



आपकी सफलता का साथी

SAD

SYSTEM ANALYSIS & DESIGN

COMPLETE STUDY MATERIAL



Easy
Language



MCQs with
Answers



Concepts with
Examples



Diagrams &
UML Basics



Exam
Focused



Complete
Coverage

Created By

♦ MARUDHARA EXAM ♦



www.marudharaexam.in



RSSB BCI

(Basic Computer Instructor)

SAD

(System Analysis & Design)

विषय सूची (INDEX)

अध्याय	विषय
Chapter 1	सिस्टम विश्लेषण एवं डिजाइन का परिचय
Chapter 2	SDLC
Chapter 3	Feasibility Study
Chapter 3(2)	System Planning
Chapter 4	DFD
Chapter 6	Requirement Analysis
Chapter 7	Decision Tree Chapter
Chapter 8	Decision Table
Chapter 9	Structured Analysis
Chapter 10	Structured Design
Chapter 11	Input Design
Chapter 12	Output Design
Chapter 13	Database Design
Chapter 14	System Testing
Chapter 15	System Implementation
Chapter 16	System Maintenance
Chapter 17	CASE Tools
Chapter 18	UML Basics
Chapter 19	MCQs

MARUDHARA EXAM

राजस्थान प्रतियोगी परीक्षाओं की सम्पूर्ण तैयारी हेतु



Handwritten
Notes



Daily
Sujaas



Daily
Quiz



PDF
Notes



RSSB & RPSC
Study Material



Website:
www.marudharaexam.in

Created By:

MARUDHARA EXAM



★ आपकी सफलता का साथी ★

मरुधरा एग्जाम

हमारा लक्ष्य - आपकी सफलता



RSSB BCI (Basic Computer Instructor)

SAD (System Analysis & Design)

विषय - SAD

हस्तलिखित नोट्स

By - मरुधरा एग्जाम

अध्याय - 1 : सिस्टम विश्लेषण एवं डिजाइन का परिचय

★ 1. परिचय

- किसी भी संगठन या संस्था में कार्यो को प्रभावी ढंग से संपन्न करने के लिए एक अच्छा सिस्टम (System) आवश्यक होता है।
- सिस्टम विश्लेषण एवं डिजाइन (SAD) वह प्रक्रिया है जिसमें किसी समस्या को समझकर, उसके समाधान के लिए एक उपयुक्त सुचना तंत्र (Information System) का विकास किया जाता है।
- SAD दो चरणों में विभाजित है -
 1. सिस्टम विश्लेषण (System Analysis)
 2. सिस्टम डिजाइन (System Design)

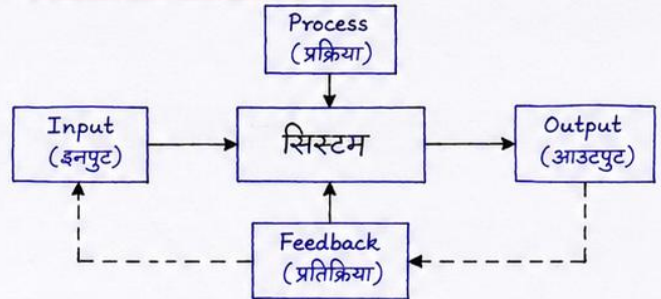
★ 2. परिभाषा

- **सिस्टम** : दो या दो से अधिक घटकों का समूह जो किसी उद्देश्य को प्राप्त करने के लिए परस्पर कार्य करते हैं, सिस्टम कहलाते हैं।
- **सिस्टम विश्लेषण** : यह मौजूदा सिस्टम का अध्ययन करके उसकी कमजोरियों, आवश्यकताओं तथा सुधार के क्षेत्रों की पहचान करने की प्रक्रिया है।
- **सिस्टम डिजाइन** : यह विश्लेषण के आधार पर एक नए सिस्टम की संरचना, मॉड्यूल, इनपुट, आउटपुट, डेटा तथा प्रक्रियाओं को निर्धारित करने की प्रक्रिया है।

★ 3. सिस्टम की विशेषताएँ

1. **उद्देश्यपरक (Objective)** - प्रत्येक सिस्टम का एक निश्चिन उद्देश्य होता है।
2. **घटक (Components)** - सिस्टम अनेक घटकों से मिलकर बना होता है।
3. **परस्पर संबंधितता (Interrelated)** - सभी घटक एक-दूसरे से जुड़े होते हैं।
4. **समग्रता (Holistic)** - सिस्टम को एक इकाई के रूप में देखा जाता है।
5. **वातावरण से संबंध (Environment)** - प्रत्येक सिस्टम अपने बाहरी वातावरण से प्रभावित होता है।

★ 4. सिस्टम के प्रमुख तत्व



इनपुट → प्रोसेस → आउटपुट
और आउटपुट की प्रतिक्रिया (Feedback) पुनः सिस्टम को सुधारने में सहायता करती है।

★ 5. सिस्टम विश्लेषण एवं डिजाइन का महत्व

- यह संगठन के कार्यो को अधिक प्रभावी एवं कुशल बनाता है।
- संसाधनों का सर्वोत्तम उपयोग सुनिश्चित करता है।
- निर्णय लेने में सहायता करता है।
- समय, लागत तथा मानवीय प्रयास की बचत होती है।
- प्रतिस्पर्धा में वृद्धि एवं बेहतर सेवाएँ प्रदान करने में मदद करता है।
- परिवर्तन को अपनाने में सहायक होता है।

ध्यान रखने योग्य बातें

- SAD का लक्ष्य समस्या का समाधान निकालकर एक उत्तम सुचना तंत्र विकसित करना है।
- अच्छा सिस्टम वही है जो उपयोगकर्ता की आवश्यकताओं को पूरा करे और संगठन के उद्देश्यों को प्राप्त करने में मदद करे।

★ मरुधरा एग्जाम - आपकी सफलता के लिए समर्पित ★

Join Us : / Marudhara Exam ★

अध्याय - 2 : SDLC (System Development Life Cycle)

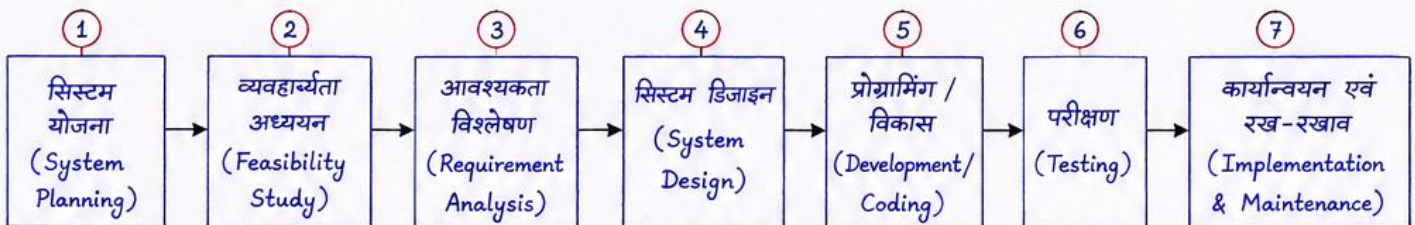
1. परिचय

- SDLC का पूरा नाम System Development Life Cycle है।
- यह एक क्रमबद्ध प्रक्रिया है, जिसका उपयोग किसी सूचना प्रणाली (Information System) को विकसित करने के लिए किया जाता है।
- SDLC यह सुनिश्चित करता है कि सिस्टम उपयोगकर्ता की आवश्यकताओं के अनुसार, उचित समय, लागत और बेहतरीन गुणवत्ता के साथ विकसित किया जाए।

2. SDLC के उद्देश्य

- सिस्टम विकास की एक तार्किक एवं वैज्ञानिक प्रक्रिया प्रदान करना।
- सिस्टम विकास में समय, लागत और प्रयास की बचत करना।
- उच्च गुणवत्ता एवं विश्वसनीय सिस्टम का विकास करना।
- परिवर्तनों को प्रभावी ढंग से प्रबंधित करना।
- उपयोगकर्ता की आवश्यकताओं को पूरा करना।
- सिस्टम के रख-रखाव (Maintenance) को आसान बनाना।

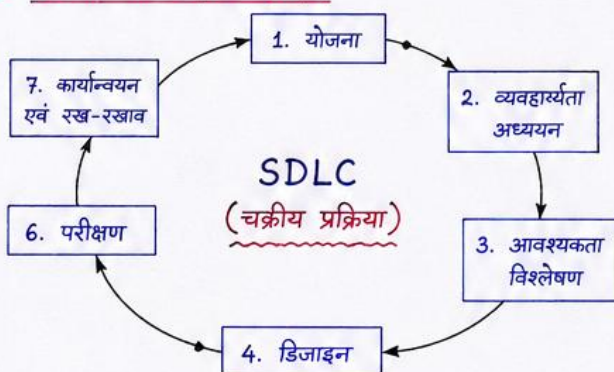
3. SDLC के चरण (Phases)



4. SDLC के चरणों का विवरण

- सिस्टम योजना (System Planning)** : इस चरण में संगठन की समस्याओं, अवसरों और उद्देश्यों की पहचान की जाती है।
- व्यवहार्यता अध्ययन (Feasibility Study)** : प्रस्तावित सिस्टम तकनीकी, आर्थिक, परिचालन और समय की दृष्टि से व्यवहार्य है या नहीं, इसका अध्ययन किया जाता है।
- आवश्यकता विश्लेषण (Requirement Analysis)** : उपयोगकर्ताओं से जानकारी एकत्र करके उनकी आवश्यकताओं को समझा जाता है और सिस्टम की आवश्यकताओं को दस्तावेज़ित किया जाता है।
- सिस्टम डिजाइन (System Design)** : सिस्टम के डेटा, प्रक्रियाओं, इंटरफेस, आउटपुट, डेटाबेस आदि का डिजाइन तैयार किया जाता है।
- प्रोग्रामिंग / विकास (Development / Coding)** : डिजाइन के आधार पर प्रोग्राम कोड लिखे जाते हैं और सिस्टम का विकास किया जाता है।
- परीक्षण (Testing)** : विकसित सिस्टम को विभिन्न परीक्षणों के द्वारा उसकी त्रुटियों को खोजकर सुधार किया जाता है।
- कार्यान्वयन एवं रख-रखाव (Implementation & Maintenance)** : सिस्टम को वास्तविक वातावरण में लागू किया जाता है और समय-समय पर रख-रखाव एवं सुधार किया जाता है।

5. SDLC का चक्रीय स्वरूप



6. SDLC के लाभ

- यह प्रक्रिया व्यवस्थित एवं तार्किक होती है।
- समय और लागत की बचत होती है।
- उच्च गुणवत्ता वाला सिस्टम विकसित होता है।
- परिवर्तनों को संभालना आसान होता है।
- दस्तावेज़ीकरण बेहतर होता है।
- रख-रखाव एवं सुधार करना सरल होता है।

नोट : SDLC कोई सख्त नियम नहीं है, इसे संगठन की आवश्यकता के अनुसार अनुकूलित (Tailor) किया जा सकता है।

अध्याय - 3 : व्यवहार्यता अध्ययन (Feasibility Study)

1. परिचय

- व्यवहार्यता अध्ययन (Feasibility Study) SDLC का दूसरा चरण है, जो यह निर्धारित करता है कि कोई प्रस्तावित प्रणाली बनाना व्यवहारिक है या नहीं।
- इस अध्ययन का उद्देश्य यह जानना है कि सीमित संसाधनों में प्रस्तावित प्रणाली को लागू करना लाभकारी होगा या नहीं।
- यदि प्रणाली व्यवहार्य पाई जाती है तभी आगे के चरणों में कार्य किया जाता है, अन्यथा परियोजना को अस्वीकार कर दिया जाता है।

2. व्यवहार्यता अध्ययन के उद्देश्य

- प्रस्तावित विनिर्माण प्रणाली की व्यवहार्यता (Practicality) का मूल्यांकन करना।
- प्रणाली के लाभ एवं खर्च का विश्लेषण करना।
- जोखिमों की पहचान करना।
- संसाधनों (मनुष्य, धन, समय, तकनीक) की उपलब्धता का मूल्यांकन करना।
- निर्णय लेना कि परियोजना को आगे बढ़ाया जाए या नहीं।

3. व्यवहार्यता अध्ययन के प्रकार

व्यवहार्यता अध्ययन के प्रकार

1. तकनीकी व्यवहार्यता (Technical Feasibility) क्या आवश्यक तकनीक, हार्डवेयर, सॉफ्टवेयर उपलब्ध है?	2. आर्थिक व्यवहार्यता (Economic Feasibility) क्या लागत उचित है? लाभ लागत से अधिक होंगे या नहीं?	3. संचालनात्मक व्यवहार्यता (Operational Feasibility) क्या प्रणाली का उपयोग उपयोगकर्ता आसानी से कर पाएंगे?	4. समय व्यवहार्यता (Time Feasibility) क्या प्रणाली को निर्धारित समय में विकसित किया जा सकेगा?	5. कानूनी व्यवहार्यता (Legal Feasibility) क्या प्रणाली सभी कानूनी नियमों और नीतियों के अनुसार है?
--	---	---	---	---

4. व्यवहार्यता अध्ययन का विस्तृत वर्णन

- तकनीकी व्यवहार्यता** : इसमें जाँच की जाती है कि क्या आवश्यक तकनीक, हार्डवेयर, सॉफ्टवेयर, विशेषज्ञता आदि उपलब्ध है या नहीं।
- आर्थिक व्यवहार्यता** : इसमें लागत और लाभ का विश्लेषण किया जाता है। लागत-लाभ विश्लेषण (Cost Benefit Analysis) किया जाता है।
- संचालनात्मक व्यवहार्यता** : क्या संगठन की वर्तमान कार्य-प्रणाली, कर्मचारी और उपयोगकर्ता को अपनाने के लिए तैयार है या नहीं।
- समय व्यवहार्यता** : क्या प्रणाली को उपलब्ध समय सीमा में विकसित और लागू किया जा सकेगा।
- कानूनी व्यवहार्यता** : क्या प्रणाली सभी कानूनी आवश्यकताओं, नीतियों, लाइसेंस, और सरकारी नियमों के अनुरूप है या नहीं।

5. व्यवहार्यता अध्ययन की प्रक्रिया

- समस्या की पहचान
- प्रस्तावित समाधान का अध्ययन
- विभिन्न व्यवहार्यता पहलुओं का विश्लेषण
- लागत एवं लाभ का आकलन
- रिपोर्ट तैयार करना
- निर्णय लेना (स्वीकृति / अस्वीकृति)

महत्वपूर्ण नोट

व्यवहार्यता अध्ययन में यदि किसी भी प्रमुख पहलू में प्रस्तावित प्रणाली व्यवहार्य नहीं पाई जाती है तो परियोजना को आगे नहीं बढ़ाया जाता है।

6. व्यवहार्यता अध्ययन के महत्वपूर्ण कारक

- | | |
|---|---|
| <ul style="list-style-type: none"> ★ संगठन के उद्देश्यों के साथ संगतता ★ तकनीक की उपलब्धता ★ मानव संसाधनों की उपलब्धता ★ लागत एवं लाभ ★ समय सीमा | <ul style="list-style-type: none"> ★ वर्तमान प्रणाली की स्थिति ★ भविष्य में विस्तार की संभावना ★ जोखिम एवं अनिश्चितताएँ ★ कानूनी एवं नीतिगत नियम ★ उपयोगकर्ता की स्वीकार्यता |
|---|---|

याद रखें

व्यवहार्यता अध्ययन किसी भी प्रणाली के सफल विकास की नींव है। यदि नींव मजबूत होगी तो प्रणाली भी सफल होगी।



मरुधरा एग्जाम

हमारा लक्ष्य - आपकी सफलता



RSSB BCI (Basic Computer Instructor)

SAD (System Analysis & Design)

विषय - SAD

हस्तलिखित नोट्स

By - मरुधरा एग्जाम

अध्याय - 3 : सिस्टम प्लानिंग (System Planning)

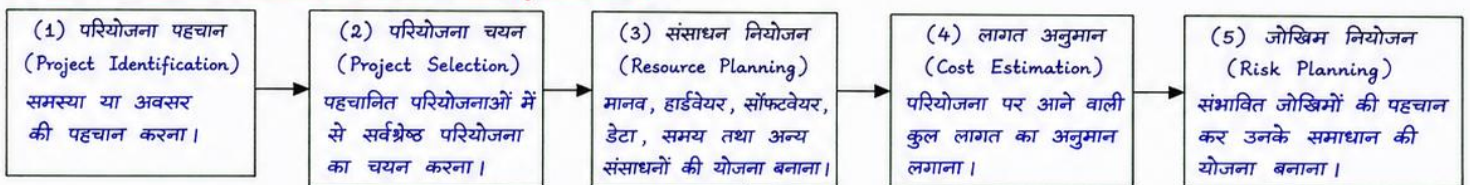
★ 1. परिचय

- सिस्टम प्लानिंग (System Planning) SDLC का प्रथम चरण है जिसमें भविष्य की आवश्यकताओं को ध्यान में रखते हुए सूचना प्रणाली के विकास की **समग्र योजना** बनाई जाती है।
- इसका उद्देश्य यह सुनिश्चित करना है कि **सही समस्या** के लिए **सही समाधान** चुना जाए और संसाधनों का सर्वोत्तम उपयोग हो।
- यह पूरे प्रोजेक्ट के लिए दिशा (Direction) प्रदान करता है।

★ 2. सिस्टम प्लानिंग का उद्देश्य

- संगठन की वर्तमान स्थिति का अध्ययन करना।
- भविष्य की आवश्यकताओं का पूर्वानुमान करना।
- सूचना प्रणाली विकास के लिए उपयुक्त रणनीति बनाना।
- संसाधनों का सही उपयोग सुनिश्चित करना।
- जोखिम (Risk) को पहचानकर न्यूनतम करना।
- परियोजना के सफलता की संभावना को अधिकतम करना।
- प्रबंधन को बेहतर निर्णय लेने में सहायता करना।

★ 3. सिस्टम प्लानिंग की प्रक्रिया (Planning Process)



4. परियोजना पहचान (Project Identification)

- संगठन की समस्याओं, अवसरों और लक्ष्यों का अध्ययन किया जाता है।
- इसके स्रोत -
 - प्रबंधन के लक्ष्य (Organizational Goals)
 - उपयोगकर्ताओं की मांग (User Requests)
 - कार्यप्रणाली में कमियाँ (Operational Problems)
 - नई तकनीक के अवसर (Technological Opportunities)
- आउटपुट** - संभावित परियोजनाओं की सूची।

5. परियोजना चयन (Project Selection)

- सभी संभावित परियोजनाओं में से सर्वोत्तम परियोजना का चयन किया जाता है।
- चयन के मानदंड (Criteria) -
 - संगठन के लक्ष्यों के साथ संगतता (Goal Compatibility)
 - लागत-लाभ अनुपात (Cost-Benefit Ratio)
 - तकनीकी व्यवहार्यता (Technical Feasibility)
 - संसाधनों की उपलब्धता (Resource Availability)
 - कार्यान्वयन की कठिनाई (Implementation Difficulty)
 - जोखिम का स्तर (Risk Level)
- आउटपुट** - चयनित परियोजना।

6. संसाधन नियोजन (Resource Planning)

- परियोजना को सफल बनाने के लिए आवश्यक संसाधनों की योजना बनाई जाती है।
- संसाधन के प्रकार -
 - मानव संसाधन (Human Resource)
 - हार्डवेयर (Hardware)
 - सॉफ्टवेयर (Software)
 - डेटा (Data)
 - समय (Time)
 - वित्त (Finance)
- आउटपुट** - संसाधनों की उपलब्धता एवं उपयोग की योजना।

7. लागत अनुमान (Cost Estimation)

- परियोजना पर आने वाली कुल लागत का अनुमान लगाना।
- लागत के प्रकार -
 - प्रारंभिक लागत (Initial Cost)
 - विकास लागत (Development Cost)
 - संचालन लागत (Operating Cost)
 - रख-रखाव लागत (Maintenance Cost)
- आउटपुट** - कुल अनुमानित लागत रिपोर्ट।

8. जोखिम नियोजन (Risk Planning)

- संभावित जोखिमों (Risks) की पहचान कर उनके प्रभाव को कम करने की योजना बनाना।
- जोखिम के प्रकार - तकनीकी, वित्तीय, मानव संसाधन, समय, प्राकृतिक, संचालन संबंधी आदि।
- आउटपुट** - जोखिम न्यूनीकरण योजना (Risk Mitigation Plan)।



सिस्टम प्लानिंग का महत्व

- यह सही समस्या और सही समाधान चुनने में सहायता करता है।
- संसाधनों का बेहतर उपयोग सुनिश्चित करता है।
- परियोजना की सफलता की संभावना बढ़ाता है।
- समय, लागत और जोखिम को नियंत्रित करने में मदद करता है।
- यह पूरे SDLC के लिए मजबूत आधार प्रदान करता है।

★ मरुधरा एग्जाम - आपकी सफलता के लिए समर्पित ★

3

Join Us : / Marudhara Exam ★

अध्याय - 4 : DFD (Data Flow Diagram)

1. परिचय

- DFD (Data Flow Diagram) एक ग्राफिकल तकनीक है, जो सिस्टम में डेटा के प्रवाह (Flow) को दर्शाती है।
- यह सिस्टम के विभिन्न कार्य, बाहरी इकाइयों, डेटा स्टोरेज और उनके बीच डेटा के प्रवाह को दिखाती है।
- DFD सिस्टम विश्लेषण (System Analysis) का एक महत्वपूर्ण टूल है।
- यह लॉजिक मॉडल (Logic Model) को दर्शाती है, फिजिकल डायग्राम को नहीं।

2. DFD के उद्देश्य

1. सिस्टम में डेटा के स्रोत और गंतव्य को दर्शाना।
2. डेटा के प्रवाह और उसकी दिशा को दिखाना।
3. सिस्टम के प्रोसेस (कार्य) को समझाना।
4. बाहरी इकाइयों एवं डेटा स्टोरेज की पहचान करना।
5. सिस्टम की जटिलता को सरल रूप में प्रदर्शित करना।
6. सिस्टम डिज़ाइन के लिए आधार तैयार करना।

3. DFD के प्रकार

1. भौतिक DFD (Physical DFD)
 - यह सिस्टम के भौतिक पहलुओं को दर्शाता है।
2. लॉजिकल DFD (Logical DFD)
 - यह सिस्टम के लॉजिक को दर्शाता है।

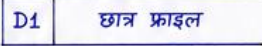
4. DFD के स्तर (Levels)

1. संदर्भ आरेख (Context Diagram / Level-0)
2. स्तर-0 DFD (High Level DFD)
3. स्तर-1 DFD (Detailed DFD)
4. स्तर-2 DFD (और अधिक विस्तृत)

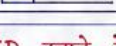
नोट :

DFD को टॉप-डाउन दृष्टिकोण (Top-Down Approach) से बनाया जाता है।

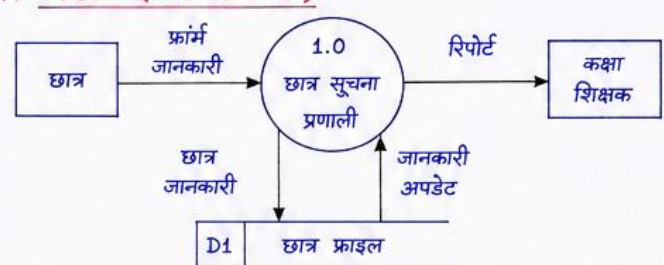
5. DFD के घटक (Components)

(i) बाहरी इकाई (External Entity) वे व्यक्ति, संगठन या सिस्टम जो सिस्टम को डेटा देते हैं या सिस्टम से डेटा प्राप्त करते हैं। उदाहरण - छात्र, ग्राहक, बैंक आदि। 	(ii) प्रक्रिया (Process) यह वह कार्य है जो डेटा को इनपुट के रूप में लेकर आउटपुट के रूप में देता है। उदाहरण - डेटा जाँचना, गणना करना, रिपोर्ट बनाना आदि। 	(iii) डेटा प्रवाह (Data Flow) यह तीर (Arrow) के रूप में दर्शाया जाता है जो डेटा की दिशा को दिखाता है। उदाहरण - छात्र विवरण 	(iv) डेटा स्टोरेज (Data Store) यह डेटा को संग्रहित करने के स्थान को दर्शाता है। उदाहरण - फ़ाइल, डेटाबेस, टेबल आदि। 
--	--	--	---

6. DFD के प्रतीक (Symbols)

क्रा.	प्रतीक	नाम	विवरण
1.		बाहरी इकाई	सिस्टम के बाहर की इकाई जो डेटा देती या प्राप्त करती है।
2.		प्रक्रिया	डेटा को इनपुट के रूप में लेकर आउटपुट प्रदान करने वाला कार्य।
3.		डेटा प्रवाह	डेटा की दिशा और प्रवाह को दर्शाता है।
4.		डेटा स्टोरेज	डेटा को संग्रहित करने का स्थान।

7. उदाहरण (स्तर-0 DFD)



8. DFD बनाने के नियम

1. DFD में डेटा प्रवाह को तीर (Arrow) से दर्शाएँ।
2. प्रत्येक प्रक्रिया का कम से कम एक इनपुट और एक आउटपुट होता है।
3. डेटा प्रवाह हमेशा प्रक्रिया से जुड़ा होना चाहिए, कभी भी प्रक्रिया के बीच सीधे डेटा प्रवाह नहीं होगा।
4. बाहरी इकाई का सीधे डेटा स्टोरेज से संबंध नहीं होता।
5. DFD को ऊपर से नीचे (Top-Down) की ओर विकसित करें।
6. प्रत्येक DFD के लिए एक संदर्भ आरेख अवश्य बनाएं।

9. महत्वपूर्ण बिंदु

- ★ DFD सिस्टम विश्लेषण का महत्वपूर्ण उपकरण है।
- ★ DFD लॉजिक मॉडल को दर्शाता है, फिजिकल नहीं।
- ★ DFD को सरल से जटिल (Level-0 से Level-n) की ओर बनाते हैं।
- ★ प्रतीकों का मानक उपयोग DFD को समझने में आसान बनाता है।
- ★ DFD सिस्टम डिज़ाइन और प्रोग्रामिंग के लिए आधार प्रदान करता है।

10. अभ्यास प्रश्न (MCQs)

1. DFD का पूरा नाम क्या है? (A) Data File Diagram (B) Data Flow Diagram (C) Data Flow Design (D) Data Form Diagram उत्तर - (B)	2. DFD में प्रक्रिया को किस प्रतीक से दर्शाया जाता है? (A) आयत (B) वृत्त (C) तीर (D) समानांतर रेखाएँ उत्तर - (B)	3. डेटा स्टोरेज को दर्शाने वाला प्रतीक कौन सा है? (A) आयत (C) समानांतर रेखाएँ (D) तीर उत्तर - (C)	4. बाहरी इकाई का उदाहरण है - (A) प्रोसेस (B) फ़ाइल (C) छात्र (D) डेटा प्रवाह उत्तर - (C)
---	---	---	---

अध्याय - 6 : आवश्यकता विश्लेषण (Requirement Analysis)

1. परिचय

- आवश्यकता विश्लेषण (Requirement Analysis) वह प्रक्रिया है जिसमें उपयोगकर्ता की आवश्यकताओं, समस्याओं और अपेक्षाओं को समझकर उन्हें स्पष्ट रूप से परिभाषित किया जाता है।
- यही प्रक्रिया सफल प्रणाली विकसित करने की नींव होती है।
- यदि आवश्यकताओं को सही से नहीं समझा गया तो प्रणाली उपयोगी नहीं होगी और समय तथा धन की हानि होगी।
- यह SDLC का महत्वपूर्ण चरण है जो System Planning और Design के बीच आता है।

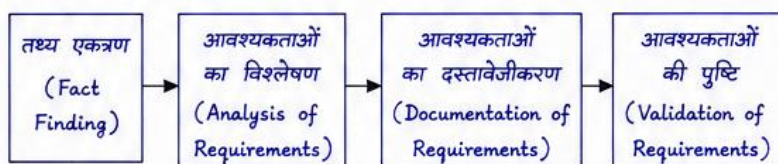
2. आवश्यकता विश्लेषण के उद्देश्य

- सही आवश्यकताओं को पहचानना और दस्तावेज़ करना।
- उपयोगकर्ता (Users) की समस्याओं को समझना।
- व्यवहार्य (Feasible) और व्यावहारिक समाधान प्रदान करना।
- आवश्यकताओं को प्राथमिकता (Priority) देना।
- डिज़ाइन और विकास के लिए स्पष्ट आधार प्रदान करना।
- भविष्य में परिवर्तन करने में सहायक होना।

3. आवश्यकता विश्लेषण की विशेषताएँ

- ★ उपयोगकर्ता केंद्रित (User Oriented)
- ★ वास्तविक और पूर्ण (Realistic & Complete)
- ★ सुस्पष्ट और समझने योग्य (Clear & Understandable)
- ★ सुसंगत (Consistent)
- ★ परिवर्तनीय (Modifiable)
- ★ परीक्षण योग्य (Verifiable)
- ★ प्राथमिकता आधारित (Priority Based)

4. आवश्यकता विश्लेषण की प्रक्रिया (समग्र दृष्टि)



 **नोट :** आवश्यकताओं का सही और पूर्ण दस्तावेजीकरण भविष्य में होने वाले विवादों और परिवर्तन की लागत को कम करता है।

5. तथ्य एकत्रण (Fact Finding Techniques)

(1) साक्षात्कार (Interview)	(2) प्रश्नावली (Questionnaire)	(3) अवलोकन (Observation)	(4) दस्तावेज़ अध्ययन (Document Review)	(5) सर्वेक्षण (Survey)	(6) सैंपलिंग (Sampling)
					
विशेषज्ञ व्यक्ति से आगने-सामने बात कर के जानकारी एकत्र करना।	लिखित प्रश्नों की सूची देकर जानकारी प्राप्त करना।	प्रणाली / कार्य को प्रत्यक्ष रूप से देखकर जानकारी लेना।	मौजूदा दस्तावेज़ों / रिपोर्ट / फॉर्म आदि का अध्ययन करना।	जानकारी हेतु व्यापक स्तर पर सर्वे करना।	समूह के कुछ प्रतिनिधि सदस्यों से जानकारी लेना।
लाभ : • गहराई से जानकारी मिलती है। • जटिल प्रश्न पूछे जा सकते हैं।	लाभ : • एक साथ अधिक लोगों से जानकारी मिलती है। • रिकॉर्ड रखना आसान।	लाभ : • वास्तविक स्थिति का पता चलता है। • छिपी हुई जानकारी मिलती है।	लाभ : • पहले से उपलब्ध जानकारी मिलती है। • कम समय और कम लागत।	लाभ : • व्यापक और विविध जानकारी। • निर्णय में सहायक।	लाभ : • समय और लागत कम। • बड़े समूह का अनुमान मिलता है।
हानि : • समय अधिक लगता है। • महंगा (Costly) होता है।	हानि : • सभी प्रश्नों के उत्तर नहीं मिलते। • गलत समझने की संभावना।	हानि : • समय अधिक लगता है। • पर्यवेक्षक की पक्षपातिता हो सकती है।	हानि : • पुरानी या अपूर्ण जानकारी मिल सकती है।	हानि : • समय और लागत अधिक। • सर्वे का विस्तार बढ़ा होता है।	हानि : • नमूना सही न हो तो परिणाम भ्रामक हो सकते हैं।

6. आवश्यकता विश्लेषण में एकत्र की जाने वाली जानकारी

- संगठन की सामान्य जानकारी (Organization Information)
- वर्तमान प्रणाली (Current System) का विवरण
- समस्याएँ (Problems) और उनके कारण
- उपयोगकर्ता की आवश्यकताएँ (User Requirements)
- इनपुट, आउटपुट, प्रोसेसिंग और नियंत्रण से संबंधित जानकारी
- प्रदर्शन (Performance) आवश्यकताएँ
- सुरक्षा (Security) आवश्यकताएँ
- भविष्य की संभावित आवश्यकताएँ

 **नोट :** जितनी अधिक स्पष्ट और सही जानकारी एकत्र की जाएगी, उतना ही बेहतर डिज़ाइन और प्रणाली विकसित होगी।

7. अच्छी आवश्यकताओं की विशेषताएँ

S Specific (विशिष्ट)	→ आवश्यकताएँ स्पष्ट और निश्चित होनी चाहिए।
M Measurable (मापने योग्य)	→ आवश्यकताओं को मापा और परखा जा सके।
A Achievable (प्राप्त करने योग्य)	→ आवश्यकताएँ व्यावहारिक और संभव होनी चाहिए।
R Relevant (संबंधित)	→ आवश्यकताएँ संगठन के लक्ष्यों से संबंधित हों।
T Time-bound (समयबद्ध)	→ आवश्यकताओं को निर्धारित समय सीमा में पूरा किया जा सके।



अध्याय - 7 : निर्णय वृक्ष (Decision Tree)

1. परिचय

- निर्णय वृक्ष (Decision Tree) एक ग्राफिकल तकनीक है, जो जटिल निर्णयों को सरल रूप में दर्शाती है।
- यह तकनीक किसी समस्या या स्थिति में विभिन्न विकल्पों (Alternatives) और उनके परिणामों (Outcomes) को वृक्ष के रूप में प्रदर्शित करती है।
- इसका प्रयोग प्रायः ऐसी परिस्थितियों में किया जाता है जहाँ भविष्य के परिणाम अनिश्चित (Uncertain) होते हैं।
- यह "क्या होगा यदि" (What if) प्रकार के निर्णय लेने में सहायक होती है।

2. निर्णय वृक्ष (Decision Tree) के प्रतीक (Symbols)

क्रम.	प्रतीक	नाम	विवरण
1.		निर्णय बिंदु (Decision Node)	यह वर्ग (□) किसी निर्णय को दर्शाता है जहाँ एक से अधिक विकल्प होते हैं।
2.		परिणाम बिंदु (Outcome Node)	यह वृत्त (○) किसी परिणाम या घटना को दर्शाता है।
3.		शाखा / लिंक (Branch / Link)	यह रेखा (—) निर्णय बिंदु को परिणाम बिंदु से जोड़ती है।
4.		लागत / संभावना (Cost / Probability)	शाखाओं पर लिखी जाने वाली लागत या संभाव्यता को दर्शाता है।

3. निर्णय वृक्ष के घटक (Components)

- निर्णय बिंदु (Decision Node) - वर्ग (□) के रूप में, जहाँ विकल्प उपलब्ध होते हैं।
- परिणाम बिंदु (Outcome Node) - वृत्त (○) के रूप में, जो संभावित परिणामों को दर्शाते हैं।
- शाखाएँ (Branches) - रेखाएँ जो निर्णय को परिणाम से जोड़ती हैं।
- विकल्प (Alternatives) - निर्णय बिंदु से निकलने वाले मार्ग।
- लागत / लाभ (Cost / Benefit) - प्रत्येक परिणाम से जुड़ी लागत या लाभ।
- संभावना (Probability) - किसी परिणाम के घटित होने की संभाव्यता (0 से 1 के बीच)।

4. निर्णय वृक्ष की विशेषताएँ

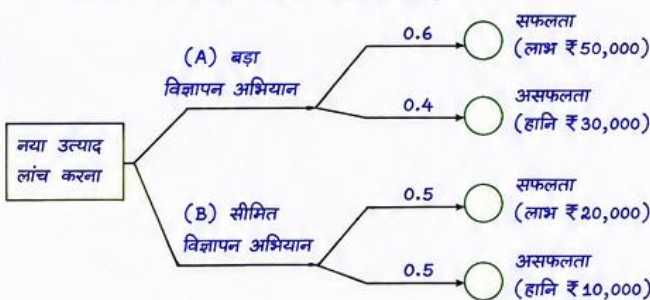
- ★ यह निर्णय करने के लिए सभी विकल्पों और उनके परिणामों को क्रमबद्ध रूप में दर्शाती है।
- ★ जटिल समस्याओं को सरल और स्पष्ट बनाती है।
- ★ प्रत्येक विकल्प के संभावित परिणाम और उनके प्रभाव को दर्शाती है।
- ★ यह निश्चितता के साथ अनिश्चितता को भी समझने में सहायक है।
- ★ लागत, लाभ और संभाव्यता के आधार पर सर्वोत्तम निर्णय चुनने में मदद करती है।



नोट : निर्णय वृक्ष विशेष रूप से प्रबंधन, वित्त, बैंकिंग, बीमा, और परियोजना चयन जैसे क्षेत्रों में बहुत उपयोगी है।

5. उदाहरण - 1 (Example)

प्रश्न : एक कंपनी नया उत्पाद बाजार में लाना चाहती है। दो विकल्प हैं - (A) बड़ा विज्ञापन अभियान (B) सीमित विज्ञापन अभियान। प्रत्येक विकल्प के संभावित परिणाम निम्न प्रकार हैं -



नोट : ऊपर 0.6, 0.4, 0.5, 0.5 संभाव्यता (Probability) दर्शाती है।

6. अपेक्षित मान (Expected Value) निकालना

$$\text{अपेक्षित मान (EMV)} = \sum (\text{संभाव्यता} \times \text{परिणाम का मान})$$

$$\text{विकल्प (A) का EMV} = (0.6 \times 50,000) + (0.4 \times (-30,000)) \\ = 30,000 - 12,000 = ₹ 18,000$$

$$\text{विकल्प (B) का EMV} = (0.5 \times 20,000) + (0.5 \times (-10,000)) \\ = 10,000 - 5,000 = ₹ 5,000$$

निष्कर्ष :

चूंकि विकल्प (A) का अपेक्षित मान (₹18,000) विकल्प (B) (₹5,000) से अधिक है, इसलिए कंपनी को (A) बड़ा विज्ञापन अभियान चुनना चाहिए।

7. निर्णय वृक्ष बनाने के चरण (Steps to Draw Decision Tree)

- समस्या को ध्यानपूर्वक समझें।
- निर्णय बिंदु (□) बनाएँ और उपलब्ध सभी विकल्प लिखें।
- प्रत्येक विकल्प से संभावित परिणाम (○) की पहचान करें।
- परिणामों की संभाव्यता (Probability) लिखें (0 से 1 के बीच)।
- प्रत्येक परिणाम से जुड़ा लागत / लाभ लिखें (₹ में)।
- अपेक्षित मान (EMV) की गणना करें और सर्वोत्तम विकल्प चुनें।



महत्वपूर्ण टिप्स

- सभी संभावित विकल्प और परिणाम अवश्य शामिल करें।
- संभाव्यता का योग हमेशा 1 होना चाहिए।
- लागत को ऋणात्मक (-) और लाभ को धनात्मक (+) संख्या में लिखें।
- निर्णय सबसे अधिक अपेक्षित लाभ (EMV) वाले विकल्प का लें।



अध्याय - 8 : निर्णय सारणी (Decision Table)

1. परिचय

- निर्णय सारणी (Decision Table) एक तालिका (Table) है जिसका उपयोग जटिल निर्णय नियमों को व्यवस्थित रूप में दर्शाने के लिए किया जाता है।
- इसमें सभी संभावित शर्तों (Conditions) के संयोजन और प्रत्येक संयोजन के लिए की जाने वाली क्रियाएँ (Actions) को दिखाया जाता है।
- यह तकनीक विशेष रूप से व्यावसायिक नियम, नीतियाँ, दरें, छूट, अनुमोदन नियम आदि को स्पष्ट करने में उपयोगी है।
- यह निर्णय लेने की प्रक्रिया को सरल, स्पष्ट और त्रुटि-रहित बनाती है।

2. निर्णय सारणी की संरचना

भाग	विवरण	उदाहरण			
शर्तें (Conditions)	सभी इनपुट स्थितियों / शर्तों जो निर्णय को प्रभावित करती हैं।	आयु > 18 आयु <= 18			
स्थिति संयोजन (Condition Stubs)	सभी शर्तों के सभी संभावित संयोजन (T/F या Y/N)।	1 T T	2 T F	3 F T	4 F F
क्रियाएँ (Actions)	प्रत्येक स्थिति संयोजन के लिए की जाने वाली क्रियाएँ।	छूट दें छूट न दें आवेदन अस्वीकार करें			
कार्य संयोजन (Action Entries)	दर्शाता है कि कौन-सी क्रिया किस स्थिति संयोजन में होगी (✓ या X द्वारा)।	1 ✓ X X	2 X ✓ X	3 ✓ X X	4 X X ✓

3. निर्णय सारणी बनाने की प्रक्रिया

- निर्णय स्थिति से संबंधित सभी शर्तों (Conditions) की सूची बनाएं।
- प्रत्येक शर्त के सभी संभावित मान निर्धारित करें (आमतौर पर 2: True/False)।
- सभी शर्तों के सभी संभावित संयोजन (Condition Stubs) बनाएं।
- निर्णय स्थिति में होने वाली सभी क्रियाएँ (Actions) की सूची बनाएं।
- प्रत्येक स्थिति संयोजन के लिए लागू क्रियाओं को चिह्नित करें (✓ = क्रिया होगी, X = क्रिया नहीं होगी)।

4. निर्णय सारणी के लाभ

- जटिल निर्णय नियमों को सरल एवं व्यवस्थित रूप में दर्शाती है।
- सभी संभावित स्थितियों (Combinations) की जाँच हो जाती है।
- निर्णय तर्क में त्रुटि की संभावना कम करती है।
- सिस्टम विकास, परीक्षण और रखरखाव में सहायक होती है।
- व्यागान्याग (Communication) को स्पष्ट बनाती है।

5. निर्णय सारणी का घटक विवरण

- Conditions (शर्तें) : इनपुट स्थितियों जो निर्णय को प्रभावित करती हैं।
- Condition Stubs (स्थिति संयोजन) : सभी शर्तों के सभी संभव संयोजन।
- Actions (क्रियाएँ) : निर्णय के परिणामस्वरूप की जाने वाली क्रियाएँ।
- Action Entries (कार्य संयोजन) : कौन-सी क्रिया किस स्थिति संयोजन पर लागू होगी।

6. प्रतीक (Symbols)

प्रतीक	अर्थ	उपयोग
T (True)	सत्य	शर्त सत्य है / पूरी होती है
F (False)	असत्य	शर्त असत्य है / पूरी नहीं होती
✓	हाँ / होगा	क्रिया उस संयोजन में होगी
X	नहीं / नहीं होगा	क्रिया उस संयोजन में नहीं होगी
-	कोई प्रभाव नहीं	उस शर्त का उस संयोजन पर प्रभाव नहीं

7. उदाहरण (Example)

- प्रश्न : एक लाइब्रेरी में पुस्तक निर्गमन (Issue) के लिए नियम निम्न हैं -
- (C1) सदस्यता वैध है (Valid Membership) या नहीं
- (C2) सदस्य पर कोई बकाया राशि है (Dues Pay) या नहीं
- नियम :
- (R1) यदि सदस्यता वैध है और कोई बकाया नहीं है → पुस्तक निर्गमन करें।
- (R2) यदि सदस्यता वैध नहीं है → पुस्तक निर्गमन न करें।
- (R3) यदि बकाया है → बकाया राशि जमा कराएँ, फिर पुस्तक निर्गमन करें।

शर्तें (Conditions)	स्थिति संयोजन (Condition Stubs)			
	1	2	3	4
C1 : सदस्यता वैध है	T	T	F	F
C2 : बकाया राशि नहीं है	T	F	T	F
क्रियाएँ (Actions)	कार्य संयोजन (Action Entries)			
A1 : पुस्तक निर्गमन करें	✓	X	X	X
A2 : बकाया राशि जमा कराएँ	X	✓	X	X
A3 : पुस्तक निर्गमन न करें	X	X	✓	✓

8. उदाहरण की व्याख्या

- स्थिति 1 (C1=T, C2=T) → सदस्यता वैध और कोई बकाया नहीं → A1 होगी (पुस्तक निर्गमन करें)
- स्थिति 2 (C1=T, C2=F) → सदस्यता वैध पर बकाया है → A2 होगी (पहले बकाया जमा कराएँ, फिर पुस्तक दें)
- स्थिति 3 (C1=F, C2=T) → सदस्यता वैध नहीं, चाहे बकाया हो या न हो → A3 होगी (पुस्तक निर्गमन न करें)
- स्थिति 4 (C1=F, C2=F) → सदस्यता वैध नहीं और बकाया भी है → A3 होगी (पुस्तक निर्गमन न करें)



नोट : • निर्णय सारणी में 2^n संयोजन बनते हैं, जहाँ n = शर्तों (Conditions) की संख्या।
• जैसे 3 शर्तें हों तो $2^3 = 8$ संयोजन बनेंगे।



अध्याय - 9 : संरचित विश्लेषण (Structured Analysis)

1. परिचय

- संरचित विश्लेषण (Structured Analysis) एक प्रणाली (System) का अध्ययन करने की एक तकनीक है, जिसमें समस्या को छोटे-छोटे भागों (Modules) में बाँटकर समझा जाता है।
- इसका उद्देश्य सिस्टम की आवश्यकताओं को स्पष्ट रूप से समझना और तार्किक मॉडल (Logical Model) बनाना है।
- यह विश्लेषण मुख्यतः "कार्य" (Function) पर आधारित होता है, ना कि "डेटा" पर।
- संरचित विश्लेषण में DFD (Data Flow Diagram) मुख्य उपकरण के रूप में उपयोग किया जाता है।

2. संरचित विश्लेषण के उद्देश्य

- वर्तमान प्रणाली (Existing System) को समझना।
- समस्याओं और कमियों की पहचान करना।
- उपयोगकर्ता की आवश्यकताओं को स्पष्ट और पूर्ण रूप से परिभाषित करना।
- सिस्टम की कार्यात्मक आवश्यकताओं को दस्तावेज़ित करना।
- सिस्टम के तार्किक मॉडल का निर्माण करना।
- आगे के डिजाइन (Design) और विकास (Development) के लिए आधार तैयार करना।

3. संरचित विश्लेषण की मुख्य विशेषताएँ

- ★ यह प्रक्रियात्मक (Process Oriented) होता है।
- ★ DFD के माध्यम से सिस्टम का मॉडल बनाया जाता है।
- ★ टॉप-डाउन दृष्टिकोण (Top-Down Approach) अपनाया जाता है।
- ★ कार्यात्मक अपघटन (Functional Decomposition) किया जाता है।
- ★ डेटा और नियंत्रण प्रवाह (Data & Control Flow) को स्पष्ट किया जाता है।
- ★ सरल, समझने में आसान और दस्तावेज़ीकरण योग्य होता है।
- ★ बड़े और जटिल सिस्टम को भी सरल भागों में बाँटता है।

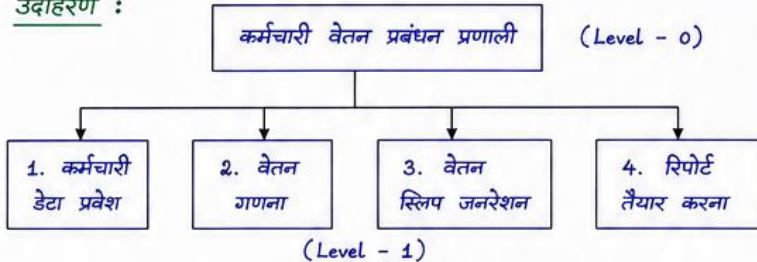
4. संरचित विश्लेषण के प्रमुख उपकरण (Tools)

- डेटा फ्लो डायग्राम (Data Flow Diagram - DFD)
- डेटा डिक्शनरी (Data Dictionary)
- प्रोसेस स्पेसिफिकेशन (Process Specification)
- मिनी-स्पेस (Mini-Spec)
- डेटा स्टोर डिक्शनरी (Data Store Dictionary)
- स्ट्रक्चर्ड इंग्लिश / फ्यूडो कोड (Structured English / Pseudo Code)

5. कार्यात्मक अपघटन (Functional Decomposition)

- किसी जटिल कार्य (Complex Function) को समझने के लिए उसे छोटे-छोटे उप-कार्य (Sub-Functions) में बाँटा जाता है।
- यह प्रक्रिया टॉप-डाउन (Top-Down) तरीके से की जाती है।

उदाहरण :



नोट :

- किसी भी प्रणाली को छोटे-छोटे कार्यों में इस प्रकार बाँटा जाता है ताकि प्रत्येक कार्य को अलग से समझा और मॉडल किया जा सके।
- इससे सिस्टम का विश्लेषण आसान और प्रभावी हो जाता है।

6. DFD और संरचित विश्लेषण का संबंध

- संरचित विश्लेषण का मुख्य उपकरण DFD है।
- DFD के विभिन्न स्तर (Levels) निम्न प्रकार होते हैं :

लेवल	नाम	विवरण
Level - 0	Context Diagram	पूरे सिस्टम का समग्र दृश्य
Level - 1	Top Level DFD	प्रमुख प्रक्रियाएँ और उनके डेटा फ्लो
Level - 2	Detailed DFD	प्रत्येक प्रक्रिया का विस्तार
Level - n	Further Detail	आवश्यकतानुसार और अधिक विस्तार

7. संरचित विश्लेषण की प्रक्रिया (Steps)

1. समस्या की पहचान और अध्ययन
2. तथ्य संग्रह (Fact Finding)
3. सिस्टम की आवश्यकताओं को निर्धारित करना
4. DFD बनाना (Level 0 से निम्न स्तरों तक)
5. प्रोसेस स्पेसिफिकेशन लिखना
6. डेटा डिक्शनरी तैयार करना
7. सिस्टम मॉडल की समीक्षा और सत्यापन करना

8. संरचित विश्लेषण के लाभ

- ★ जटिल प्रणाली को सरल बनाता है।
- ★ प्रणाली को समझना आसान होता है।
- ★ ब्रुटियों और समस्याओं की पहचान आसान होती है।
- ★ उपयोगकर्ता की आवश्यकताएँ स्पष्ट होती हैं।
- ★ सिस्टम डिजाइन और विकास के लिए मजबूत आधार प्रदान करता है।
- ★ दस्तावेज़ीकरण पूरा और व्यवस्थित होता है।
- ★ संचार (Communication) में सुधार होता है।



अध्याय - 10 : संरचित विज्ञान (Structured Design)

1. परिचय

- संरचित डिज़ाइन (Structured Design) वह प्रक्रिया है जिसमें सिस्टम की आवश्यकताओं और विश्लेषण के परिणामों को ध्यान में रखते हुए एक ऐसा डिज़ाइन तैयार किया जाता है जो **प्रभावी, कुशल, समझने में आसान और रखरखाव योग्य** हो।
- यह चरण Structured Analysis के बाद और Coding (Programming) से पहले आता है।
- इसका उद्देश्य सिस्टम को **मॉड्यूल (Modules)** में बाँटकर उनका विस्तृत डिज़ाइन तैयार करना होता है।

2. संरचित विज्ञान के उद्देश्य

- सिस्टम को **मॉड्यूल (Modules)** में विभाजित करना।
- प्रत्येक मॉड्यूल का विस्तृत डिज़ाइन तैयार करना।
- मॉड्यूल के बीच के संबंध (Interfaces) को स्पष्ट करना।
- डेटा, प्रोसेसिंग और नियंत्रण की स्पष्ट संरचना बनाना।
- सिस्टम को **प्रभावी, पुनः उपयोग योग्य (Reusable)** और रखरखाव योग्य बनाना।
- प्रोग्रामिंग (Coding) को आसान और व्यवस्थित बनाना।

3. संरचित विज्ञान की विशेषताएँ

- ★ यह **मॉड्यूल आधारित (Module Oriented)** होता है।
- ★ इसे **Top-Down** या **Bottom-Up** तरीके से किया जा सकता है।
- ★ इसमें **डेटा और प्रोसेसिंग** का स्पष्ट विवरण होता है।
- ★ यह **समझने, विकसित करने और रखरखाव करने में आसान** होता है।
- ★ यह **पुनः उपयोग (Reusability)** को बढ़ावा देता है।
- ★ यह **त्रुटियों (Errors)** को कम करता है।

4. संरचित विज्ञान के प्रमुख सिद्धान्त (Principles)

- मॉड्यूलरिटी (Modularity)** - सिस्टम को छोटे-छोटे मॉड्यूल में बाँटना।
- एबस्ट्रैक्शन (Abstraction)** - केवल आवश्यक जानकारी पर ध्यान देना, अनावश्यक विवरण छिपाना।
- परतबद्धता (Layering)** - सिस्टम को परतों (Layers) में संगठित करना।
- सूचना छुपाव (Information Hiding)** - मॉड्यूल का आंतरिक विवरण छिपाना।
- उच्च सामंजस्य (High Cohesion)** - एक मॉड्यूल में संबंधित कार्य ही हों।
- कम कपलिंग (Low Coupling)** - मॉड्यूल के बीच निर्भरता न्यूनतम हो।

5. मॉड्यूल (Module)

- मॉड्यूल सिस्टम का एक स्वतंत्र भाग होता है जो एक या अधिक कार्य (Functions) करता है।
- प्रत्येक मॉड्यूल का एक स्पष्ट उद्देश्य होता है।
- मॉड्यूल के प्रकार :
 - स्वतंत्र मॉड्यूल (Independent Module)
 - आवृत्त मॉड्यूल (Dependent Module)
 - मुख्य मॉड्यूल (Main Module)
 - उपयोगी मॉड्यूल (Utility Module)

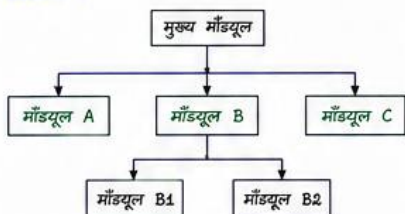
6. मॉड्यूल के गुण

गुण (Property)	विवरण
1. सामंजस्य (Cohesion)	एक मॉड्यूल के अंदर के कार्य आपस में संबंधित होने चाहिए।
2. कपलिंग (Coupling)	एक मॉड्यूल का दूसरे मॉड्यूल पर निर्भरता कम होनी चाहिए।
3. आकार (Size)	मॉड्यूल का आकार न बहुत बड़ा हो न बहुत छोटा।
4. सरलता (Simplicity)	मॉड्यूल सरल और समझने में आसान होना चाहिए।
5. पुनः उपयोगिता (Reusability)	मॉड्यूल को भविष्य में पुनः उपयोग किया जा सके।

7. डिज़ाइन की रणनीतियाँ (Design Strategies)

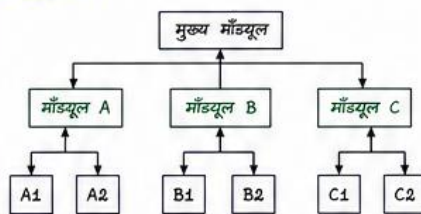
(i) टॉप-डाउन डिज़ाइन (Top-Down Design)

- इसमें पहले मुख्य मॉड्यूल को डिज़ाइन किया जाता है फिर उसे छोटे-छोटे मॉड्यूल में विभाजित किया जाता है।
- यह विश्लेषण से डिज़ाइन की ओर जाता है।
- उदाहरण :



(ii) बॉटम-अप डिज़ाइन (Bottom-Up Design)

- इसमें पहले छोटे-छोटे मॉड्यूल बनाए जाते हैं फिर उन्हें जोड़कर बड़े मॉड्यूल बनाए जाते हैं।
- यह निर्माण की ओर से डिज़ाइन की ओर जाता है।
- उदाहरण :



8. मॉड्यूल के बीच का इंटरफेस (Interface)

- इंटरफेस वह बिंदु है जहाँ दो मॉड्यूल आपस में संचार करते हैं।
- यह डेटा, नियंत्रण संकेत या जानकारी का आदान-प्रदान हो सकता है।
- अच्छा इंटरफेस मॉड्यूल के बीच निर्भरता को कम करता है।
- इंटरफेस के प्रकार :
 - पैरामीटर पासिंग (Parameter Passing)
 - डेटा कॉमन ब्लॉक (Common Data)
 - कंट्रोल फ्लैग (Control Flag)
 - मैसेज पासिंग (Message Passing)

9. उच्च सामंजस्य और कम कपलिंग

उच्च सामंजस्य (High Cohesion)

- मॉड्यूल के अंदर के कार्य आपस में संबंधित होते हैं।
- उच्च सामंजस्य अच्छा होता है।
- उदाहरण : वेतन गणना मॉड्यूल (जिसमें केवल वेतन से संबंधित कार्य हों)

कम कपलिंग (Low Coupling)

- मॉड्यूल के बीच निर्भरता कम होती है।
- कम कपलिंग अच्छा होता है।
- उदाहरण : यदि एक मॉड्यूल में परिवर्तन हो तो दूसरे मॉड्यूल प्रभावित न हों।

💡 नोट : उच्च सामंजस्य और कम कपलिंग ही अच्छे डिज़ाइन की कुंजी है।

10. संरचित विज्ञान की प्रक्रिया (Steps in Structured Design)

1	आवश्यकताओं और विश्लेषण के परिणामों की समीक्षा करना।
2	सिस्टम को प्रमुख मॉड्यूल में विभाजित करना।
3	प्रत्येक मॉड्यूल के कार्य (Functions) को निर्धारित करना।
4	मॉड्यूल के इनपुट, आउटपुट, प्रोसेसिंग और डेटा को डिज़ाइन करना।
5	मॉड्यूल के बीच इंटरफेस (Interface) को परिभाषित करना।
6	मॉड्यूल का विस्तृत डिज़ाइन दस्तावेज बनाना।



अध्याय - 11 : इनपुट डिजाइन (Input Design)

1. परिचय

- इनपुट डिजाइन (Input Design) वह प्रक्रिया है जिसमें सिस्टम में डेटा प्रवेश (Data Entry) के लिए इनपुट साधनों, इनपुट फॉर्म और इनपुट नियंत्रण (Input Controls) का निर्धारण किया जाता है।
- अच्छा इनपुट डिजाइन सिस्टम में सही, पूर्ण और समय पर डेटा प्रवेश को सुनिश्चित करता है।
- यह उपयोगकर्ता के लिए आसान, त्रुटि-रहित और कुशल होना चाहिए।

2. इनपुट डिजाइन के उद्देश्य

- सिस्टम में सही और पूर्ण डेटा प्रविष्टि सुनिश्चित करना।
- डेटा प्रविष्टि को सरल और उपयोगकर्ता के अनुकूल बनाना।
- इनपुट त्रुटियों को कम करना और डेटा की गुणवत्ता बढ़ाना।
- डेटा प्रोसेसिंग को तेज़ और कुशल बनाना।
- इनपुट डेटा की सुरक्षा और गोपनीयता सुनिश्चित करना।

3. इनपुट डिजाइन के घटक (Components)

- इनपुट माध्यम (Input Medium) - वह स्रोत जिससे डेटा प्रणाली में प्रवेश करता है।
- इनपुट डिवाइस (Input Device) - वह उपकरण जिससे डेटा सिस्टम में दिया जाता है।
- इनपुट फॉर्म (Input Form) - डेटा प्रविष्टि के लिए डिजाइन किया गया फॉर्म।
- इनपुट नियंत्रण (Input Controls) - डेटा प्रविष्टि की शुद्धता और विश्वसनीयता सुनिश्चित करने के उपाय।
- इनपुट डेटा (Input Data) - वह डेटा जो प्रणाली में प्रोसेस करने के लिए दिया जाता है।

4. इनपुट माध्यम (Input Medium)

डेटा विभिन्न स्रोतों से प्रणाली में प्रवेश करता है -

क्रम.	इनपुट माध्यम	विवरण	उदाहरण
1.	कागज़ आधारित	कागज़ पर लिखे डेटा से	फॉर्म, एप्लीकेशन, रजिस्टर
2.	इलेक्ट्रॉनिक	इलेक्ट्रॉनिक माध्यम के द्वारा	की-बोर्ड, स्कैनर, ट्रांजेक्शन
3.	ऑनलाइन	नेटवर्क / इंटरनेट के द्वारा	वेब फॉर्म, ऑनलाइन ऑर्डर
4.	मशीन जनित	किसी अन्य सिस्टम द्वारा जनित	सेंसर डेटा, मशीन आउटपुट

5. इनपुट डिवाइस (Input Devices)

डेटा प्रविष्टि के लिए प्रयुक्त प्रमुख उपकरण -

- की-बोर्ड (Keyboard)
- माउस (Mouse)
- स्कैनर (Scanner)
- बारकोड रीडर (Barcode Reader)
- OMR (Optical Mark Reader)
- MICR (Magnetic Ink Character Reader)
- टच स्क्रीन (Touch Screen)
- वेब कैमरा / बायोमेट्रिक डिवाइस आदि



6. इनपुट फॉर्म (Input Form) के प्रकार

इनपुट फॉर्म को निम्न प्रकार से वर्गीकृत किया जाता है -

- मैन्युअल फॉर्म (Manual Form) - हाथ से भरे जाने वाले फॉर्म।
- मशीन रीडेबल फॉर्म (Machine Readable Form) - जिन्हें मशीन द्वारा पढ़ा जा सके, जैसे OMR, MICR।
- ऑनलाइन फॉर्म (Online Form) - वेब या मोबाइल ऐप पर भरे जाने वाले फॉर्म।
- प्री-प्रिंटेड फॉर्म (Pre-printed Form) - पहले से छपे हुए फॉर्म जिनमें स्थायी जानकारी पहले से होती है।



नोट : अच्छा फॉर्म सरल, स्पष्ट, पूर्ण और उपयोगकर्ता के अनुकूल होना चाहिए। फॉर्म का लेआउट तार्किक और व्यवस्थित होना चाहिए।

7. इनपुट नियंत्रण (Input Controls)

इनपुट नियंत्रण डेटा की शुद्धता, पूर्णता और वैधता सुनिश्चित करने के उपाय हैं -

- वैधता नियंत्रण (Validation Control) - डेटा की वैधता जांचना।
- सीमा जांच (Range Check) - डेटा निर्धारित सीमा के भीतर है या नहीं।
- प्रारूप जांच (Format Check) - डेटा का प्रारूप सही है या नहीं।
- उपस्थिति जांच (Presence Check) - सभी अनिवार्य फ़िल्ड भरे गए हैं या नहीं।
- पूर्णता जांच (Completeness Check) - सभी आवश्यक जानकारी पूरी है या नहीं।
- संगतता जांच (Consistency Check) - डेटा में तार्किक संगतता है या नहीं।
- डुप्लिकेट जांच (Duplicate Check) - डेटा दोहराया तो नहीं गया।
- चेक डिजिट (Check Digit) - गणना द्वारा त्रुटि की जांच करना (जैसे क्रेडिट कार्ड नंबर, MICR कोड आदि)।

8. अच्छे इनपुट डिजाइन के दिशानिर्देश (Guidelines)

- फॉर्म सरल, स्पष्ट और समझने में आसान होना चाहिए।
- उपयोगकर्ता के दृष्टिकोण से फॉर्म का डिजाइन करें।
- अनावश्यक जानकारी के लिए स्थान न दें।
- फ़िल्ड का आकार उपयुक्त रखें (छोटा न हो, बहुत बड़ा न हो)।
- तार्किक क्रम में जानकारी रखें (ऊपर से नीचे, बाएँ से दाएँ)।
- महत्वपूर्ण निर्देश और उदाहरण दें।
- त्रुटि संदेश स्पष्ट और उपयोगी होने चाहिए।
- इनपुट डेटा की गोपनीयता और सुरक्षा सुनिश्चित करें।

9. इनपुट डिजाइन को प्रभावित करने वाले कारक



10. इनपुट फॉर्म का उदाहरण

कर्मचारी जानकारी फॉर्म	
कर्मचारी आई.डी. :	दिनांक : [/ /]
नाम :	
पता :	
मोबाइल नंबर :	
विभाग :	पद :
जन्म तारीख : [/ /]	ईमेल :

नोट : * चिह्नित फ़िल्ड भरना अनिवार्य है।





अध्याय - 12 : आउटपुट डिज़ाइन (Output Design)

1. परिचय

- आउटपुट डिज़ाइन (Output Design) वह प्रक्रिया है जिसमें सिस्टम से प्राप्त होने वाली सुचनाओं (Outputs) को उपयोगकर्ता के लिए उपयोगी, स्पष्ट, सटीक और समय पर प्रस्तुत किया जाता है।
- इसका उद्देश्य सुचना को इस रूप में प्रस्तुत करना है कि वह उपयोगकर्ता की आवश्यकताओं को पूरा कर सके।
- अच्छा आउटपुट प्रबंधन निर्णय लेने, नियंत्रण और योजना बनाने में सहायक होता है।

2. आउटपुट डिज़ाइन के उद्देश्य

- उपयोगकर्ता की आवश्यकताओं के अनुसार आउटपुट प्रस्तुत करना।
- सुचना को स्पष्ट, सटीक, समय पर और पूर्ण बनाना।
- महत्वपूर्ण सुचना को उपयुक्त रूप में प्रदर्शित करना।
- आउटपुट की फॉर्मेटिंग और संरचना को मानकीकृत करना।
- आउटपुट की पठनीयता और समझ को बढ़ाना।
- अनावश्यक सुचना को हटाकर प्रासंगिक सुचना प्रदान करना।
- आउटपुट की सुरक्षा और गोपनीयता सुनिश्चित करना।

3. आउटपुट के प्रकार (Types of Outputs)

- हार्ड कॉपी आउटपुट (Hard Copy Output) - प्रिंटर/ग्राफिक्स डिवाइस द्वारा कागज़ पर प्राप्त आउटपुट।
उदा. : रिपोर्ट, बिल, इनवॉइस, पेरोल शीट आदि।
- सॉफ्ट कॉपी आउटपुट (Soft Copy Output) - मॉनिटर पर प्रदर्शित आउटपुट।
उदा. : स्क्रीन रिपोर्ट, ग्राफ, चार्ट आदि।
- ग्राफिकल आउटपुट (Graphical Output) - चार्ट, ग्राफ, डायग्राम के रूप में प्रस्तुत आउटपुट।
- मल्टीमीडिया आउटपुट (Multimedia Output) - ध्वनि, वीडियो, एनीमेशन आदि के रूप में।

4. आउटपुट डिज़ाइन के घटक (Components)

- आउटपुट की सामग्री (Content) - कौन-सी सुचना प्रदर्शित करनी है।
- आउटपुट का प्रारूप (Format) - सुचना का लेआउट, क्रम और संरचना।
- आउटपुट माध्यम (Medium) - किस उपकरण द्वारा आउटपुट दिया जाएगा।
- आउटपुट का समय (Timing) - कब आउटपुट जनरेट करना है।
- आउटपुट की आवृत्ति (Frequency) - कितनी बार आउटपुट देना है।
- आउटपुट की गंतव्य (Destination) - आउटपुट किसे और कहाँ भेजना है।
- आउटपुट नियंत्रण (Output Controls) - वैधता, पूर्णता और सुरक्षा सुनिश्चित करना।

5. आउटपुट माध्यम (Output Medium)

क्रम.	माध्यम	विवरण	उदाहरण
1.	मॉनिटर (VDU)	स्क्रीन पर आउटपुट प्रदर्शित करना	स्क्रीन रिपोर्ट, चार्ट
2.	प्रिंटर	कागज़ पर आउटपुट प्रिंट करना	रिपोर्ट, बिल, स्टेटमेंट
3.	प्लॉटर	बड़े चार्ट, ग्राफ आदि बनाना	इंजीनियरिंग ड्राइंग, चार्ट
4.	स्पीकर	ध्वनि के रूप में आउटपुट प्रदान करना	अलर्ट साउंड, संदेश
5.	प्रोजेक्टर	स्क्रीन/दीवार पर प्रदर्शन करना	प्रेजेंटेशन, स्लाइड्स

6. आउटपुट का प्रारूप (Format)

एक सानात्य रिपोर्ट का प्रारूप

कंपनी का नाम / लोगो
रिपोर्ट का शीर्षक
दिनांक : __ / __ / __ समय : __ : __
रिपोर्ट की अवधि : _____

क्रम.	विवरण	मात्रा	दर	राशि
1.			--	
2.			--	
3.			--	
कुल योग :				

तैयार किया : _____ स्वीकृत : _____ पृष्ठ-संख्या : ____

- (i) हेडर
→ (ii) रिपोर्ट की जानकारी
→ (iii) कॉलम हेडिंग
→ (iv) विवरण क्षेत्र
→ (v) टोटल / समरी
→ (vi) फुटर

7. आउटपुट डिज़ाइन की विशेषताएँ

- स्पष्ट और सरल भाषा में होना चाहिए।
- सही और सटीक सुचना प्रदान करनी चाहिए।
- समय पर उपलब्ध होना चाहिए।
- उपयोगकर्ता की आवश्यकताओं के अनुसार होना चाहिए।
- सुगठित, आकर्षक और पठनीय होना चाहिए।
- अनावश्यक विवरण से मुक्त होना चाहिए।
- सुरक्षा और गोपनीयता सुनिश्चित होनी चाहिए।
- त्रुटि रहित और पूर्ण होना चाहिए।

8. आउटपुट डिज़ाइन की प्रक्रिया (Steps)

- उपयोगकर्ता की आवश्यकताओं का अध्ययन करना।
- आवश्यक आउटपुट की पहचान करना।
- आउटपुट की सामग्री और प्रारूप तय करना।
- आउटपुट माध्यम का चयन करना।
- आउटपुट का लेआउट और संरचना तैयार करना।
- नमूना आउटपुट (Sample Output) बनाना और परीक्षण करना।
- अंतिम आउटपुट को लागू एवं उपयोग में लाना।

9. आउटपुट के लिए दिशानिर्देश (Guidelines)

- केवल उपयोगी और प्रासंगिक सुचना ही प्रदर्शित करें।
- सुचना को तार्किक क्रम (Logical Order) में व्यवस्थित करें।
- जहाँ आवश्यक हो, ग्राफ, चार्ट, टेबल आदि का प्रयोग करें।
- डेटा की इकाइयों (Units) और लेबल स्पष्ट लिखें।
- हेडर और फुटर में आवश्यक जानकारी दें (जैसे - दिनांक, समय, पृष्ठ संख्या आदि)।
- त्रुटियों को कम करने हेतु वैधता जाँच (Validation) करें।

10. अच्छे आउटपुट डिज़ाइन के लाभ

- निर्णय लेने में सहायता मिलती है।
- कार्य की निगरानी और नियंत्रण आसान होता है।
- समय और लागत की बचत होती है।
- उपयोगकर्ता की संतुष्टि बढ़ती है।
- संगठन की उत्पादकता और दक्षता में वृद्धि होती है।
- सुचना का प्रभावी उपयोग सुनिश्चित होता है।



नोट : आउटपुट वह दर्पण है, जिसमें सिस्टम की सफलता उपयोगकर्ता को दिखाई देती है।



अध्याय - 13 : डेटाबेस डिजाइन (Database Design)

1. परिचय

- डेटाबेस डिजाइन वह प्रक्रिया है जिसमें किसी संगठन की जानकारी को सुव्यवस्थित तरीके से संग्रहित करने के लिए डेटाबेस की संरचना तैयार की जाती है।
- यह डेटा (Data) को प्रभावी, सुरक्षित और तेज़ी से उपयोग योग्य बनाने में मदद करता है।
- डेटाबेस डिजाइन का मुख्य उद्देश्य डेटा की पुनरावृत्ति कम करना, डेटा की अखंडता बनाए रखना और डेटा को आसानी से प्राप्त करना है।

3. Database Design के स्तर (Levels)

- Conceptual Design (अवधारणात्मक स्तर)
 - पूरे सिस्टम का समग्र डेटाबेस संरचना डिजाइन किया जाता है।
 - ER Model (Entity Relationship Model) का उपयोग।
- Logical Design (तार्किक स्तर)
 - अवधारणात्मक डिजाइन को किसी डेटा मॉडल में परिवर्तित किया जाता है जैसे - Relational Model, Hierarchical Model आदि।
- Physical Design (भौतिक स्तर)
 - वास्तविक भंडारण (Storage) के लिए डेटाबेस की संरचना बनाई जाती है।
 - फ़ाइल संगठन, इंडेक्सिंग, एक्सेस मेथड आदि तय किए जाते हैं।

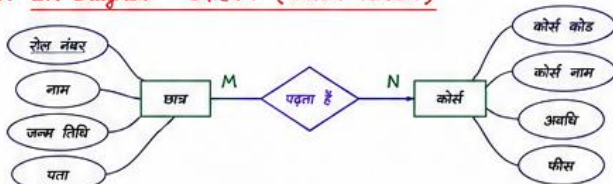
5. ER Model (Entity Relationship Model) क्या है?

- ER Model एक अवधारणात्मक डेटा मॉडल है जिसे Peter Chen ने प्रस्तुत किया।
- यह व्यावहारिक दुनिया की वस्तुओं (Entities) और उनके बीच संबंधों (Relationships) को चित्रात्मक रूप में प्रस्तुत करता है।
- ER Model तीन मूल घटकों पर आधारित है - Entity, Attribute और Relationship।

7. Attribute (गुण / विशेषता)

- Entity की विशेषताओं को Attribute कहते हैं।
- प्रत्येक Attribute किसी Entity के बारे में विस्तार से जानकारी देता है।
- Attribute को Oval (दीर्घवृत्त) से दर्शाया जाता है।
- Attribute के प्रकार -
 - Simple Attribute - जो अविभाज्य हो (जैसे - नाम, आयु)
 - Composite Attribute - जो कई भागों में विभाजित हो (जैसे - पता → घर नं., शहर, राज्य, पिनकोड)
 - Single Valued Attribute - जिसका मान एक हो (जैसे - रोल नंबर)
 - Multi Valued Attribute - जिसका मान एक से अधिक हो सकता है (जैसे - फोन नंबर)
 - Derived Attribute - जिसे अन्य Attributes से निकाला जा सके (जैसे - आयु (जन्म तिथि से))
 - Key Attribute - जो किसी Entity की पहचान करे (जैसे - रोल नंबर)

9. ER Diagram - उदाहरण (कालेज सिस्टम)



नोट : M : N संबंध को व्यावहारिक डेटाबेस (Relational Model) में हल करने के लिए एक अलग तालिका (Junction Table) बनाई जाती है।

11. डेटाबेस डिजाइन के सिद्धांत (Principles)

- डेटा की आवश्यकता को समझें।
- सुरक्षा और गोपनीयता सुनिश्चित करें।
- न्यूनतम पुनरावृत्ति रखें।
- सिस्टम की स्केलेबिलिटी और लचीलापन रखें।
- डेटा की अखंडता बनाए रखें।
- बेहतर प्रदर्शन (Performance) का ध्यान रखें।
- डेटा को स्वतंत्रता दें (Data Independence)

2. Database Design के उद्देश्य

- डेटा को संगठित (Organized) रूप में संग्रहित करना।
- डेटा की पुनरावृत्ति (Redundancy) को न्यूनतम करना।
- डेटा की अखंडता (Integrity) और शुद्धता बनाए रखना।
- डेटा को तेज़ी और आसानी से प्राप्त और अपडेट करना।
- डेटा सुरक्षा (Data Security) और गोपनीयता सुनिश्चित करना।
- डेटा को साझा (Share) करने और स्केलेबिलिटी (Scalability) बढ़ाना।

4. डेटाबेस डिजाइन में उपयोग होने वाले मॉडल

- Hierarchical Model (हायरार्किकल मॉडल)
 - डेटा को पेड़ (Tree) जैसी संरचना में संग्रहित किया जाता है।
- Network Model (नेटवर्क मॉडल)
 - डेटा को ग्राफ (Graph) संरचना में संग्रहित किया जाता है।
- Relational Model (रिलेशनल मॉडल)
 - डेटा को तालिकाओं (Tables) के रूप में संग्रहित किया जाता है। (यह सबसे अधिक उपयोग किया जाने वाला मॉडल है)
- Object-Oriented Model (ऑब्जेक्ट-ओरिएंटेड मॉडल)
 - डेटा को ऑब्जेक्ट के रूप में संग्रहित और प्रबंधित किया जाता है।

6. Entity (सत्ता / इकाई)

- वास्तविक दुनिया की वह वस्तु या चीज़ जिसके बारे में जानकारी संग्रहित की जाती है, उसे Entity कहते हैं।
- जैसे - छात्र, कर्मचारी, पुस्तक, ग्राहक आदि।
- Entity को Rectangle (आयत) से दर्शाया जाता है।

उदाहरण :



8. Relationship (संबंध)

- दो या दो से अधिक Entities के बीच के संबंध को Relationship कहते हैं।
- Relationship को Diamond (हीरा) से दर्शाया जाता है।
- Relationship के प्रकार -
 - One to One (1 : 1) - एक Entity का संबंध दूसरी एक Entity से।
 - One to Many (1 : N) - एक Entity का संबंध कई Entities से।
 - Many to One (N : 1) - कई Entities का संबंध एक Entity से।
 - Many to Many (M : N) - कई Entities का संबंध कई Entities से।

उदाहरण (1 : N)



नोट : Relationship को उचित नाम (Verb) दिया जाता है जैसे - रखता है, करता है, पढ़ता है, जारी करता है आदि।

10. Normalization (नॉर्मलाइजेशन)

- Normalization वह प्रक्रिया है जिसमें तालिकाओं (Tables) को इस प्रकार डिजाइन किया जाता है कि डेटा की पुनरावृत्ति कम हो और डेटा की अखंडता बनी रहे।
- मुख्य नॉर्मल फॉर्म -
 - 1NF (First Normal Form) - प्रत्येक कॉलम में Atomic Value (अविभाज्य) हो।
 - 2NF (Second Normal Form) - 1NF में और हर Non-Key Attribute पूरी Primary Key पर निर्भर हो।
 - 3NF (Third Normal Form) - 2NF में और कोई Non-Key Attribute किसी दूसरे Non-Key Attribute पर निर्भर न हो।

12. डेटाबेस डिजाइन के लाभ

- डेटा सुव्यवस्थित और सुसंगठित रहता है।
- तेज़ और कुशल डेटा एक्सेस प्राप्त होता है।
- डेटा की पुनरावृत्ति और त्रुटियाँ कम होती हैं।
- भंडारण स्थान की बचत होती है।
- डेटा की सुरक्षा और अखंडता बढ़ती है।
- सिस्टम का रख-रखाव आसान होता है।



अध्याय - 14 : सिस्टम टेस्टिंग (System Testing)

1. परिचय

- सिस्टम टेस्टिंग वह प्रक्रिया है जिसमें विकसित सिस्टम/सॉफ्टवेयर की गुणवत्ता, कार्यक्षमता और विश्वसनीयता की जाँच की जाती है।
- इसका उद्देश्य यह सुनिश्चित करना होता है कि सिस्टम आवश्यकताओं के अनुसार सही तरीके से कार्य कर रहा है या नहीं।
- टेस्टिंग SDLC का महत्वपूर्ण चरण है जो डेवलपमेंट के बाद और इम्प्लीमेंटेशन से पहले किया जाता है।
- यह बग्स (Errors/Bugs) का पता लगाने, उन्हें ठीक करने और सिस्टम की गुणवत्ता सुधारने में मदद करती है।

3. टेस्टिंग के प्रकार (Types of Testing)

क्रम.	टेस्टिंग का प्रकार	विवरण
1.	यूनिट टेस्टिंग (Unit Testing)	प्रोग्राम के व्यक्तिगत मॉड्यूल/यूनिट की जाँच।
2.	इंटीग्रेशन टेस्टिंग (Integration Testing)	मॉड्यूल के बीच इंटरफेस और एकीकृत कार्य की जाँच।
3.	सिस्टम टेस्टिंग (System Testing)	पूरे सिस्टम की आवश्यकताओं के अनुसार जाँच।
4.	एक्सेप्टेंस टेस्टिंग (Acceptance Testing)	उपयोगकर्ता द्वारा सिस्टम को स्वीकार करने की जाँच।
5.	परफॉर्मंस टेस्टिंग (Performance Testing)	सिस्टम की गति, लोड, स्थिरता की जाँच।
6.	सिक्योरिटी टेस्टिंग (Security Testing)	सिस्टम की सुरक्षा कमजोरियों की जाँच।
7.	यूजेबिलिटी टेस्टिंग (Usability Testing)	सिस्टम की उपयोगिता और सहजता की जाँच।

5. सिस्टम टेस्टिंग के स्तर (Levels of Testing)



7. सिस्टम टेस्टिंग के प्रमुख चरण (Steps in System Testing)

- आवश्यकताओं की समीक्षा करना।
- टेस्ट प्लान तैयार करना।
- टेस्ट केस डिजाइन करना।
- टेस्ट केस को तैयार डेटा के साथ निष्पादित करना।
- वास्तविक परिणाम और अपेक्षित परिणाम की तुलना करना।
- दोषों (Defects) की पहचान करना और रिपोर्ट करना।
- दोषों को ठीक करना और पुनः टेस्टिंग करना (Retesting)।



9. टेस्टिंग मेट्रिक्स (Testing Metrics)

मेट्रिक	विवरण
टेस्ट केस की संख्या	कुल बनाए गए टेस्ट केस की संख्या
पास प्रतिशत	कुल टेस्ट में से पास हुए टेस्ट का प्रतिशत
फेल प्रतिशत	कुल टेस्ट में से फेल हुए टेस्ट का प्रतिशत
डिफेक्ट घनत्व	प्रति KLOC (हजार लाइन कोड) में दोषों की संख्या
रीवर्क प्रतिशत	कुल समय/लागत में से रीवर्क का प्रतिशत

11. अच्छी टेस्टिंग के लिए दिशानिर्देश

- टेस्टिंग की योजना जल्दी बनाएँ।
- हर जबरन और फीचर को कवर करें।
- उचित और पर्याप्त टेस्ट केस बनाएँ।
- टेस्टिंग को विभिन्न वातावरण में करें।
- दोषों को प्राथमिकता के अनुसार ठीक करें।
- री-टेस्टिंग और रेग्रेशन टेस्टिंग अवश्य करें।
- टेस्टिंग दस्तावेजों को सही और अद्यतन रखें।

नोट : अच्छी टेस्टिंग से बेहतर गुणवत्ता वाला और विश्वसनीय सिस्टम तैयार होता है।

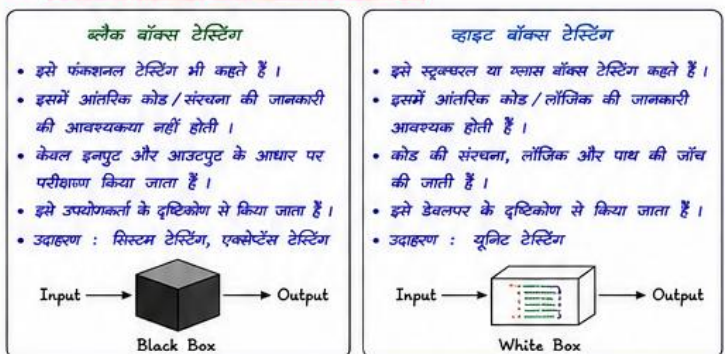
2. सिस्टम टेस्टिंग के उद्देश्य

- सिस्टम में दोष (Defects) का पता लगाना।
- यह सुनिश्चित करना कि सिस्टम आवश्यकताओं के अनुसार कार्य कर रहा है।
- सिस्टम की गुणवत्ता, विश्वसनीयता और प्रदर्शन को जाँच करना।
- संभावित समस्याओं को इम्प्लीमेंटेशन से पहले ठीक करना।
- उपयोगकर्ता को निर्बाध और संतोषजनक सिस्टम प्रदान करना।
- डेटा की सटीकता और सुरक्षा सुनिश्चित करना।

4. सिस्टम टेस्टिंग की विशेषताएँ

- यह डेवलपमेंट के बाद किया जाता है।
- यह पूरी प्रणाली (Complete System) पर किया जाता है।
- यह ब्लैक बॉक्स (Black Box) दृष्टिकोण पर आधारित होती है।
- इसमें फंक्शनल और नॉन-फंक्शनल दोनों प्रकार की जाँच होती है।
- इसका उद्देश्य सिस्टम की गुणवत्ता और उपयोगकर्ता संतुष्टि सुनिश्चित करना है।
- यह उपयोगकर्ता की आवश्यकताओं और स्पष्टीकरण के अनुसार की जाती है।

6. ब्लैक बॉक्स और व्हाइट बॉक्स टेस्टिंग



8. टेस्ट केस (Test Case)

- टेस्ट केस वह इनपुट, शर्त, परिणाम और क्रियाएँ हैं जो किसी फीचर या फंक्शन की जाँच करने के लिए उपयोग किए जाते हैं।
- एक टेस्ट केस में निम्न शामिल होते हैं -

✓ टेस्ट केस आईडी ✓ टेस्ट केस विवरण ✓ इनपुट डेटा ✓ पूर्व शर्त (Pre-condition)	✓ अपेक्षित परिणाम (Expected Result) ✓ वास्तविक परिणाम (Actual Result) ✓ टेस्ट स्थिति (Pass/Fail) ✓ टिप्पणी (Comments)
---	--

10. टेस्टिंग में उपयोग होने वाले डॉक्यूमेंट्स

- टेस्ट प्लान डॉक्यूमेंट
- टेस्ट केस डॉक्यूमेंट
- टेस्ट डेटा डॉक्यूमेंट
- टेस्ट रिपोर्ट
- डिफेक्ट रिपोर्ट
- टेस्ट समरी रिपोर्ट





अध्याय - 15 : सिस्टम इम्प्लीमेंटेशन (System Implementation)

1. परिचय

- सिस्टम इम्प्लीमेंटेशन (System Implementation) वह प्रक्रिया है जिसमें विकसित सिस्टम को वास्तविक वातावरण (Live Environment) में स्थापित कर उपयोग के लिए तैयार किया जाता है।
- इसमें हार्डवेयर, सॉफ्टवेयर, डेटाबेस, नेटवर्क, मानव संसाधन और प्रक्रियाओं का एकीकृत रूप से उपयोग किया जाता है।
- इसका मुख्य उद्देश्य - नया सिस्टम सुचारु रूप से चलाना, पुराने सिस्टम से स्थानांतरण करना और उपयोगकर्ताओं को सक्षम बनाना।

3. सिस्टम इम्प्लीमेंटेशन के प्रमुख घटक (Components)

क्रम.	घटक	विवरण	उदाहरण
1.	हार्डवेयर (Hardware)	सिस्टम के लिए आवश्यक भौतिक उपकरण	कम्प्यूटर, सर्वर, मॉडम
2.	सॉफ्टवेयर (Software)	प्रणाली संचालन हेतु प्रोग्राम	OS, DBMS, एप्लीकेशन
3.	डेटाबेस (Database)	डेटा का संग्रह, संगठन और प्रबंधन	Oracle, MySQL
4.	नेटवर्क (Network)	संसाधनों और डेटा का संचार	LAN, WAN, Internet
5.	मानव संसाधन (People)	उपयोगकर्ता, ऑपरेटर, प्रशासक	यूजर, DBA, सपोर्ट स्टाफ
6.	प्रक्रियाएँ (Procedures)	कार्य करने के नियम और निर्देश	SOP, वर्कफ्लो, नीतियाँ
7.	दस्तावेज (Documentation)	सिस्टम से संबंधित दस्तावेज	यूजर गाइड, मैनुअल, स्क्रिप्ट

5. डेटा परिवर्तन (Data Conversion) के तरीके

- डायरेक्ट कन्वर्जन (Direct Conversion)
 - सीधे पुराने डेटा को नए सिस्टम में परिवर्तित करना।
- समानांतर कन्वर्जन (Parallel Conversion)
 - पुराने और नए सिस्टम कुछ समय तक साथ-साथ चलाए जाते हैं।
- फेज्ड कन्वर्जन (Phased Conversion)
 - सिस्टम को चरणों में लागू करना (Module by Module)।
- पायलट कन्वर्जन (Pilot Conversion)
 - सीमित क्षेत्र/विभाग में सिस्टम लागू कर परीक्षण करना।
- प्लंज कन्वर्जन (Plunge Conversion)
 - पुराने सिस्टम को बंद कर एक ही समय में नया सिस्टम लागू करना।

नोट : समानांतर कन्वर्जन सबसे सुरक्षित, जबकि प्लंज कन्वर्जन सबसे तेज तरीका है।

7. सिस्टम स्थापना (System Installation) की गतिविधियाँ

- हार्डवेयर की स्थापना और कॉन्फिगरेशन
- सॉफ्टवेयर की स्थापना और सेटिंग
- डेटाबेस की स्थापना और लोडिंग
- नेटवर्क कनेक्शन की जाँच
- सुरक्षा व्यवस्थाएँ लागू करना
- प्रारंभिक डेटा लोड करना
- सिस्टम का अंतिम परीक्षण कर लाइव करना



9. इम्प्लीमेंटेशन की चुनौतियाँ और समाधान

चुनौतियाँ	समाधान
उपयोगकर्ताओं का विरोध	उन्हें समझाना और प्रशिक्षण देना
कम प्रशिक्षण	पर्याप्त और गुणवत्तापूर्ण प्रशिक्षण देना
डेटा परिवर्तन में त्रुटियाँ	डेटा सत्यापन और परीक्षण करना
हार्डवेयर/सॉफ्टवेयर समस्याएँ	पूर्व परीक्षण और गुणवत्ता जाँच
समय व बजट का अधिक होना	अच्छी योजना और नियंत्रण
संचार की कमी	प्रभावी संचार और समन्वय बनाए रखना

11. सिस्टम इम्प्लीमेंटेशन के लाभ

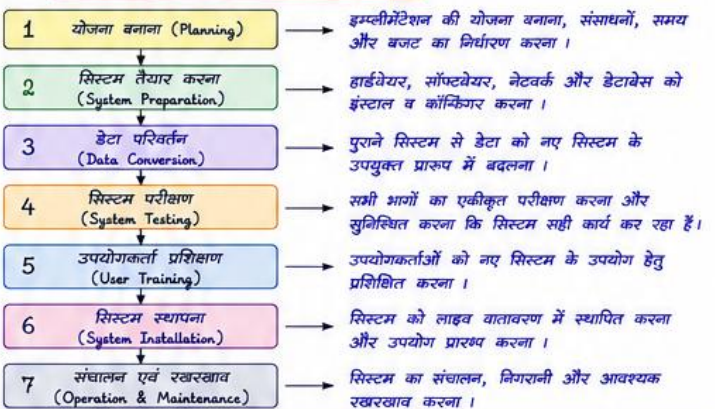
- नया सिस्टम वास्तविक वातावरण में प्रभावी रूप से कार्य करता है।
- कार्यों में गति, शुद्धता और उत्पादकता में वृद्धि होती है।
- निर्णय लेने की क्षमता में सुधार होता है।
- व्यवसायिक प्रक्रियाएँ अधिक पारदर्शी और नियंत्रित होती हैं।
- संगठन की प्रतिस्पर्धात्मक क्षमता बढ़ती है।



2. सिस्टम इम्प्लीमेंटेशन के उद्देश्य

- नया सिस्टम उपयोग के लिए तैयार करना।
- पुराने सिस्टम से डेटा और प्रक्रियाओं का सुरक्षित स्थानांतरण करना।
- उपयोगकर्ताओं को नए सिस्टम के उपयोग हेतु प्रशिक्षित करना।
- सिस्टम का परीक्षण, स्थापना और संचालन सुनिश्चित करना।
- सिस्टम के सुचारु संचालन और रखरखाव के लिए आवश्यक व्यवस्थाएँ करना।

4. सिस्टम इम्प्लीमेंटेशन के चरण (Steps)



6. उपयोगकर्ता प्रशिक्षण (User Training)

- प्रशिक्षण का उद्देश्य - उपयोगकर्ताओं को सिस्टम का प्रभावी उपयोग सिखाना।
- प्रशिक्षण के प्रकार -
 - कक्षा प्रशिक्षण (Classroom Training)
 - ऑन-द-जॉब प्रशिक्षण (On-the-Job Training)
 - ऑनलाइन प्रशिक्षण (Online Training)
 - प्रशिक्षण मैनुअल व गाइड प्रदान करना।



नोट : अच्छा प्रशिक्षण प्रणाली की सफलता में महत्वपूर्ण भूमिका निभाता है।

8. संचालन एवं रखरखाव (Operation & Maintenance)

- सिस्टम का सुचारु संचालन बनाए रखना।
- घटनाओं और त्रुटियों की निगरानी (Monitoring)।
- बैकअप और रिकवरी की व्यवस्था।
- नियमित अपडेट और सुधार करना।
- उपयोगकर्ताओं की समस्याओं का समाधान करना।
- सिस्टम सुरक्षा और गोपनीयता सुनिश्चित करना।



10. सफल इम्प्लीमेंटेशन के प्रमुख कारक (Key Success Factors)

- प्रबंधन का समर्थन
- स्पष्ट लक्ष्य और योजना
- उपयुक्त तकनीक का चयन
- उपयोगकर्ता की भागीदारी
- प्रभावी प्रशिक्षण
- अच्छी संचार व्यवस्था
- निरंतर निगरानी और समर्थन



12. सिस्टम इम्प्लीमेंटेशन की सीमाएँ

- प्रारंभिक लागत अधिक हो सकती है।
- उपयोगकर्ताओं का विरोध या अनभिज्ञता हो सकती है।
- डेटा परिवर्तन में त्रुटि की संभावना रहती है।
- स्वस्थ समस्या या तकनीकी विफलता का खतरा रहता है।
- रखरखाव और अपडेट की आवश्यकता बनी रहती है।





अध्याय - 16 : सिस्टम मेन्टीनेंस (System Maintenance)

1. परिचय

- सिस्टम मेन्टीनेंस (System Maintenance) से तात्पर्य है - स्थापित सिस्टम को समय के साथ सही, प्रभावी और अद्यतन बनाए रखना।
- यह सिस्टम के जीवन-चक्र का सबसे लंबा चरण होता है।
- समय के साथ बदलती आवश्यकताओं, तकनीक, बुटियों और सुधारों के कारण मेन्टीनेंस आवश्यक होता है।
- अच्छा मेंटेन किया गया सिस्टम अधिक विश्वसनीय, कुशल और उपयोगकर्ता के अनुकूल रहता है।

2. सिस्टम मेन्टीनेंस के उद्देश्य

- सिस्टम की बुटियों को हटकर सुधारना।
- सिस्टम को बदलती आवश्यकताओं के अनुसार अद्यतन करना।
- सिस्टम के प्रदर्शन, विश्वसनीयता और सुरक्षा को बनाए रखना।
- सिस्टम की उपयोगिता और उपयोगकर्ता संतुष्टि को बढ़ाना।
- नई तकनीक के साथ सिस्टम को सामंजस्यपूर्ण बनाए रखना।
- सिस्टम की लागत को कम करना और उपलब्धता बढ़ाना।

3. सिस्टम मेन्टीनेंस के प्रकार (Types of Maintenance)

क्रम.	प्रकार	विवरण
1.	करेक्टिव मेन्टीनेंस (Corrective)	बुटियों / दोषों को सुधारना।
2.	एडेप्टिव मेन्टीनेंस (Adaptive)	परिस्थिति (OS, हार्डवेयर, नियम, कानून आदि) में बदलाव के अनुसार संशोधन करना।
3.	परफेक्टिव मेन्टीनेंस (Perfective)	सिस्टम में सुधार, नई सुविधायें जोड़ना, प्रदर्शन बढ़ाना।
4.	प्रीवेंटिव मेन्टीनेंस (Preventive)	संभावित समस्याओं को पहले से पहचानकर उन्हें रोकना।

4. प्रकारों का विस्तृत विवरण

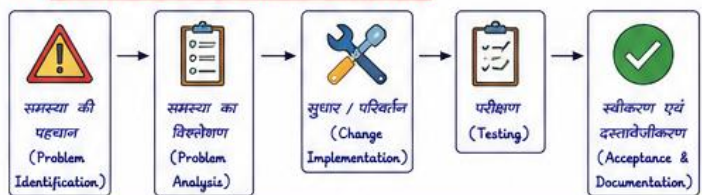
- करेक्टिव मेन्टीनेंस - जब सिस्टम में कोई बुटि पाई जाती है, तो उसे ठीक करना।
उदाहरण - प्रोग्राम में बग सुधारना, रिपोर्ट में गलत परिणाम ठीक करना।
- एडेप्टिव मेन्टीनेंस - बदलते बाहरी वातावरण के अनुसार सिस्टम में परिवर्तन करना।
उदाहरण - नया OS इंस्टॉल होने पर सॉफ्टवेयर को अपडेट करना।
- परफेक्टिव मेन्टीनेंस - सिस्टम की कार्यक्षमता, उपयोगिता और प्रदर्शन को बढ़ाना।
उदाहरण - यूजर इंटरफेस बेहतर बनाना, नई रिपोर्ट जोड़ना।
- प्रीवेंटिव मेन्टीनेंस - भविष्य की समस्याओं को रोकने के लिए कदम उठाना।
उदाहरण - नियमित बैकअप, लॉग मॉनिटरिंग, सुरक्षा पैच अपडेट।

5. परियोजना के बाद मेन्टीनेंस का महत्व

- सिस्टम की दीर्घकालिक सफलता के लिए आवश्यक।
- महंगे पुनर्निर्माण (Rebuild) की आवश्यकता को कम करता है।
- उपयोगकर्ताओं की संतुष्टि और उत्पादकता बढ़ाता है।
- सिस्टम की विश्वसनीयता और सुरक्षा सुनिश्चित करता है।
- बदलती तकनीक और व्यवसाय आवश्यकताओं के साथ तालमेल बनाए रखता है।



6. मेन्टीनेंस प्रक्रिया (Maintenance Process)



7. मेन्टीनेंस अनुरोध (Maintenance Request)

मेन्टीनेंस अनुरोध निम्न स्रोतों से प्राप्त हो सकते हैं -

- उपयोगकर्ता (User)
- प्रबंधन (Management)
- ऑपरेशन टीम (Operations Team)
- सिस्टम मॉनिटरिंग टूल (Monitoring Tools)
- कानूनी / पर्यावरणीय परिवर्तन (Legal / Environmental Change)



8. मेन्टीनेंस अनुरोध के तत्व

- | | |
|--|---|
| ☒ अनुरोध आईडी (Request ID) | ☒ प्राथमिकता (Priority) |
| ☒ अनुरोध दिनांक (Request Date) | ☒ प्रकार (Type of Maintenance) |
| ☒ अनुरोधकर्ता का नाम (Requester Name) | ☒ प्रस्तावित समाधान (Proposed Solution) |
| ☒ समस्या का विवरण (Problem Description) | ☒ स्थिति (Status) |
| ☒ प्रभावित मॉड्यूल/सिस्टम (Affected Module/System) | ☒ समाधान दिनांक (Resolution Date) |

9. मेन्टीनेंस टूल्स और तकनीक

- एरर लॉग्स (Error Logs)
- सिस्टम मॉनिटरिंग (System Monitoring)
- परफॉर्मेंस टूल (Performance Tools)
- बैकअप और रिकवरी टूल (Backup & Recovery Tools)
- वर्जन कंट्रोल सिस्टम (Version Control System)
- टिकटिंग सिस्टम (Ticketing System)
- स्वचालित अपडेट टूल (Automated Update Tools)



10. डॉक्यूमेंटेशन (Documentation) का महत्व

- सभी परिवर्तन, सुधार और परीक्षण का रिकॉर्ड रखना।
- भविष्य में समस्या निवारण में सहायक।
- नए डेवलपर/मेन्टीनेर को सिस्टम समझने में मदद।
- ज्ञान का संरक्षण (Knowledge Preservation)।



11. चुनौतियाँ (Challenges)

- बदलती तकनीक और प्लेटफॉर्म।
- अस्पष्ट / अशुद्ध आवश्यकताएं।
- पुराना (Legacy) सिस्टम।
- पर्याप्त दस्तावेज न होना।
- सीमित समय और बजट।
- कुशल मानव संसाधन की कमी।



12. अच्छी मेन्टीनेंस के लिए दिशानिर्देश

- सिस्टम का नियमित बैकअप लें।
- नियमित रूप से लॉग्स और प्रदर्शन की निगरानी करें।
- उपयोगकर्ताओं से फीडबैक नियमित रूप से लें।
- सुरक्षा अपडेट और पैच समय पर लागू करें।
- दस्तावेजों को अद्यतन और सुव्यवस्थित रखें।
- समस्याओं को प्राथमिकता के आधार पर हल करें।



13. मेन्टीनेंस मेट्रिक्स (Maintenance Metrics)

मेट्रिक	विवरण
MTTR (Mean Time To Repair)	समस्या ठीक करने में औसत समय।
MTBF (Mean Time Between Failures)	दो विफलताओं के बीच औसत समय।
Change Success Rate	सफल परिवर्तनों का प्रतिशत।
User Satisfaction	उपयोगकर्ता संतुष्टि का स्तर।
Ticket Closure Rate	बंद हुए अनुरोधों का प्रतिशत।



14. मेन्टीनेंस और विकास में अंतर

	मेन्टीनेंस (Maintenance)	विकास (Development)
उद्देश्य	मौजूदा सिस्टम को बनाए रखना	नया सिस्टम बनाना
फोकस	सुधार, संशोधन, समर्थन	विश्लेषण, डिजाइन, निर्माण
समय	दीर्घकालिक (लगातार)	सीमित (प्रोजेक्ट आधारित)
प्रयोगकर्ता की भागीदारी	मध्यम से अधिक	अधिक
जोखिम	कम से मध्यम	मध्यम से अधिक

15. सारांश

सिस्टम मेन्टीनेंस किसी भी सुचना प्रणाली के जीवन का निरंतर और महत्वपूर्ण भाग है। समय पर और प्रभावी मेन्टीनेंस से सिस्टम विश्वसनीय, सुरक्षित, अद्यतन और उपयोगकर्ता के अनुकूल बना रहता है तथा संगठन के लक्ष्यों की प्राप्ति में सहायता करता है।





अध्याय - 17 : CASE Tools (CASE उपकरण)

1. परिचय

- CASE का पूरा नाम है - Computer Aided Software Engineering.
- CASE Tools ऐसे सॉफ्टवेयर टूल्स होते हैं जो सॉफ्टवेयर विकास (Software Development) के विभिन्न चरणों में सहायता करते हैं।
- ये टूल्स उत्पादकता बढ़ाते हैं, गुणवत्ता सुधारते हैं और समय व लागत दोनों की बचत करते हैं।
- CASE Tools का उपयोग System Analysis, Design, Coding, Testing, Maintenance आदि में किया जाता है।

2. CASE Tools के उद्देश्य

- ★ सॉफ्टवेयर विकास प्रक्रिया को स्वचालित और सरल बनाना।
- ★ मानकों (Standards) के अनुसार एक समानता बनाए रखना।
- ★ डॉक्यूमेंटेशन को स्वचालित और सुसंगत (Consistent) बनाना।
- ★ डेटा, प्रक्रिया और डिजाइन के बीच एकीकरण (Integration) करना।
- ★ पुनः उपयोग (Reusability) को बढ़ावा देना।
- ★ त्रुटियों में कमी और गुणवत्ता में सुधार करना।

3. CASE Tools के प्रकार (Types of CASE Tools)

प्रकार	विवरण
1. Upper CASE Tools	यह सिस्टम विश्लेषण और डिजाइन चरण में उपयोग होते हैं। (जैसे - UML टूल्स, ERD टूल्स, DFD टूल्स आदि)
2. Lower CASE Tools	यह प्रोग्रामिंग और कोड निर्माण चरण में उपयोग होते हैं। (जैसे - IDE, Compiler, Debugger आदि)
3. Integrated CASE Tools	ये टूल्स Upper और Lower CASE दोनों की सुविधाएँ प्रदान करते हैं।
4. Component-based CASE Tools	ये टूल्स पुनः उपयोग योग्य (Reusable) घटकों (Components) के निर्माण और प्रबंधन में सहायक होते हैं।
5. Web-based CASE Tools	ये टूल्स वेब तकनीक का उपयोग करके सहयोग (Collaboration) और रिमोट एक्सेस प्रदान करते हैं।
6. Mobile CASE Tools	ये टूल्स मोबाइल प्लेटफॉर्म पर सॉफ्टवेयर विकास में सहायक होते हैं।

4. CASE Tools की विशेषताएँ

- ▶ ग्राफिकल मॉडलिंग (Graphical Modeling) प्रदान करते हैं।
- ▶ कोड जेनरेशन (Code Generation) की सुविधा देते हैं।
- ▶ डॉक्यूमेंटेशन को स्वचालित (Automated) करते हैं।
- ▶ डेटा डिक्शनरी (Data Dictionary) का समर्थन करते हैं।
- ▶ रिवर्स इंजीनियरिंग (Reverse Engineering) संभव बनाते हैं।
- ▶ प्रोजेक्ट प्रबंधन और टीम वर्क को आसान बनाते हैं।
- ▶ मानकीकरण (Standardization) को सुनिश्चित करते हैं।
- ▶ गुणवत्ता नियंत्रण और परीक्षण (Testing) को सरल बनाते हैं।

5. CASE Tools के लाभ

- ✓ उत्पादकता (Productivity) में वृद्धि
- ✓ विकास समय (Development Time) में कमी
- ✓ लागत (Cost) में कमी
- ✓ गुणवत्ता (Quality) में सुधार
- ✓ सुसंगत और मानकीकृत दस्तावेजीकरण
- ✓ सॉफ्टवेयर पुनः उपयोग (Software Reusability) में वृद्धि



6. CASE Tools की सीमाएँ

- ⊗ उच्च लागत (High Cost)
- ⊗ प्रशिक्षित और कुशल कर्मचारियों की आवश्यकता
- ⊗ टूल्स पर अधिक निर्भरता
- ⊗ छोटे प्रोजेक्ट्स के लिए अनुपयुक्त
- ⊗ टूल्स के बीच असंगतता (Incompatibility) की समस्या



7. CASE Tools के कार्य क्षेत्र (Functional Areas)

 Requirements Engineering आवश्यकताओं का संग्रह और प्रबंधन	 Analysis & Design मॉडलिंग, DFD, ERD, UML आदि	 Coding कोड जेनरेशन, एडिटिंग और कंपाइलिंग	 Testing टेस्ट केस, टेस्ट मैनेजमेंट और त्रुटि ट्रैकिंग	 Maintenance परिवर्तन प्रबंधन, दस्तावेज अपडेट और सपोर्ट
---	---	---	--	---

8. CASE Tools के उदाहरण

टूल का नाम	प्रकार	उपयोग	प्रमुख विशेषताएँ
IBM Rational Rose	Upper CASE	सिस्टम मॉडलिंग (UML)	UML डायग्राम, रिवर्स इंजीनियरिंग, कोड जेनरेशन
Microsoft Visio	Upper CASE	डायग्राम और फ्लोचार्ट बनाना	विभिन्न डायग्राम टेम्पलेट, आसान इंटरफेस
ERwin Data Modeler	Upper CASE	डेटा मॉडलिंग (DFD, ERD)	डेटा मॉडलिंग, डेटा डिक्शनरी, रिपोर्ट जेनरेशन
PowerDesigner	Upper CASE	एंटरप्राइज मॉडलिंग	UML, Business Process, डेटाबेस डिजाइन
Visual Studio	Lower CASE	कोडिंग और एप्लिकेशन विकास	IDE, डिबगर, कंपाइलर, टेस्टिंग टूल्स
Eclipse	Lower CASE	जावा/अन्य भाषाओं में विकास	ओपन सोर्स, प्लग-इन सपोर्ट, कोड एडिटिंग
JBuilder	Lower CASE	Java एप्लिकेशन विकास	GUI बिल्डर, कोड जेनरेशन, डिबगर
Together (CA)	Integrated	पूरे SDLC के लिए	Requirements, Design, Code, Test, Deploy
Oracle Designer	Upper CASE	डेटाबेस डिजाइन	डेटाबेस मॉडलिंग, फॉरवर्ड/रिवर्स इंजीनियरिंग
StarUML	Upper CASE	UML मॉडलिंग	फ्री ओपन सोर्स, UML डायग्राम सपोर्ट

9. CASE Tools और SDLC



💡 CASE Tools प्रत्येक चरण में सहायता करके SDLC को अधिक कुशल और प्रभावी बनाते हैं।

11. CASE Tools का चयन करते समय ध्यान देने योग्य बातें

- ★ प्रोजेक्ट का आकार और जटिलता
- ★ उपयोग में आसानी (Ease of Use)
- ★ समर्थित प्लेटफॉर्म और तकनीक
- ★ लागत और प्रशिक्षण आवश्यकता
- ★ उपलब्ध सुविधाएँ और कार्यक्षमता
- ★ विक्रेता (Vendor) का समर्थन और अपडेट



10. CASE Environment के घटक



12. निष्कर्ष

CASE Tools सॉफ्टवेयर विकास प्रक्रिया को व्यवस्थित, मानकीकृत और प्रभावी बनाते हैं। ये टूल्स समय, लागत और प्रयास की बचत करते हुए गुणवत्ता को बेहतर बनाते हैं। इसलिए आधुनिक सॉफ्टवेयर विकास में CASE Tools का महत्व अत्यधिक है।





अध्याय - 18 : UML Basics (UML की मूल बातें)

1. परिचय

- UML का पूरा नाम Unified Modeling Language है।
- यह एक मानक दृष्टांतक मॉडलिंग भाषा है जिसका उपयोग सॉफ्टवेयर सिस्टम के डिजाइन, विश्लेषण, निर्माण और दस्तावेजीकरण के लिए किया जाता है।
- UML को Grady Booch, James Rumbaugh और Ivar Jacobson द्वारा विकसित किया गया।
- UML को OMG (Object Management Group) द्वारा मानकीकृत किया गया है।

3. UML के डायग्राम (Types of UML Diagrams)

वर्गीकरण	डायग्राम का नाम	उद्देश्य
Structure Diagrams	1. Class Diagram	सिस्टम की संरचना और कक्षाओं के संबंध को दर्शाता है।
	2. Object Diagram	कक्षाओं के ऑब्जेक्ट्स और उनके संबंध को दिखाता है।
	3. Package Diagram	पैकेज और उनके निर्भरताओं को दर्शाता है।
	4. Component Diagram	सॉफ्टवेयर घटकों और उनके संबंध को दिखाता है।
	5. Deployment Diagram	हार्डवेयर नोड्स और सॉफ्टवेयर की तैनाती को दिखाता है।
Behavior Diagrams	6. Use Case Diagram	सिस्टम के फंक्शनल आवश्यकताओं को दर्शाता है।
	7. Activity Diagram	गतिविधि फ्लो और ब्लॉक्स को दिखाता है।
	8. Sequence Diagram	ऑब्जेक्ट्स के बीच संदेशों के क्रम को दर्शाता है।
	9. Communication Diagram	ऑब्जेक्ट्स के बीच संचार और संदेशों को दिखाता है।
	10. State Machine Diagram	किसी ऑब्जेक्ट की अवस्थाओं और ट्रांजिशन को दिखाता है।
	11. Timing Diagram	ऑब्जेक्ट्स की अवस्था परिवर्तन के साथ समय को दर्शाता है।
	12. Interaction Overview Diagram	इंटरैक्शन का उच्च-स्तरीय ओवरलू देता है।

5. Use Case Diagram के मूल तत्व

तत्व	विवरण
Actor (अभिनेता)	सिस्टम के साथ इंटरैक्ट करने वाला बाहरी उपयोगकर्ता या अन्य सिस्टम।
Use Case (उपयोगिता)	सिस्टम द्वारा प्रदान की जाने वाली सेवाएं / कार्य।
Include (समाविष्ट)	अनिवार्य रूप से एक अन्य Use Case को शामिल करना।
Extend (विस्तार)	वैकल्पिक रूप से एक अन्य Use Case का विस्तार करना।

7. Relationships (संबंधों) के प्रकार

संबंध	प्रतीक	विवरण
Association (संबंध)	—	दो कक्षाओं के बीच सामान्य संबंध।
Directed Association (निर्देशित संबंध)	→	एकतरफा संबंध।
Generalization (आभासीकरण)	—△—	एक कक्षा दूसरी की विशेष कक्षा (inheritance)।
Realization (साकारता)	---▷	Interface को implement करना।
Dependency (निर्भरता)	--->	अस्थायी उपयोग/निर्भरता।
Aggregation (समाहार)	---◇---	कमजोर "has-a" संबंध।
Composition (संघटन)	---◆---	मजबूत "part-of" संबंध।

9. Sequence Diagram के मूल तत्व

तत्व	विवरण
Lifeline	ऑब्जेक्ट का समय के साथ अस्तित्व।
Object	सिस्टम में ऑब्जेक्ट।
Message	ऑब्जेक्ट्स के बीच संदेश।
Activation	ऑब्जेक्ट का सक्रिय समय।
Return Message	संदेश का प्रत्यावर्तन।

11. UML के प्रमुख दृष्टिकोण (Views)

दृष्टिकोण (View)	विवरण
Use Case View	सिस्टम की आवश्यकताओं को दर्शाता है।
Logical View	सिस्टम की संरचना (कक्षा, संबंध आदि) को दर्शाता है।
Component View	सिस्टम के घटकों और उनके निर्भरताओं को दर्शाता है।
Deployment View	हार्डवेयर नोड्स और सॉफ्टवेयर तैनाती को दर्शाता है।
Process View	प्रक्रियाओं, समानांतरता और प्रदर्शन को दर्शाता है।

2. UML की विशेषताएँ

- मानक (Standard) और प्लेटफॉर्म स्वतंत्र (Platform Independent)
- दृष्टांतक (Visual) और समझने में आसान।
- विभिन्न दृष्टिकोणों (Views) से सिस्टम का मॉडल बनाना संभव
- विस्तृत डॉक्यूमेंटेशन और बेहतर संचार में सहायक
- पुनः उपयोग (Reusability) और रखरखाव (Maintainability) को बढ़ाता है।
- सॉफ्टवेयर विकास की सभी गतिविधियों में उपयोगी

4. UML के बिल्डिंग ब्लॉक्स (Building Blocks)

बिल्डिंग ब्लॉक	विवरण
Things (वस्तुएँ)	सिस्टम के स्थिर तत्व (जैसे - Class, Interface, Use Case, Component, Node आदि)।
Relationships (संबंध)	वस्तुओं के बीच संबंध (जैसे - Association, Dependency, Generalization आदि)।
Diagrams (डायग्राम)	वस्तुओं और संबंधों का समूह जो किसी विशेष दृष्टिकोण (View) को दर्शाता है।
Views (दृष्टिकोण)	सिस्टम का विशिष्ट परिप्रेक्ष्य (जैसे - Logical View, Component View, Deployment View)।

6. Class Diagram के मूल तत्व

खंड	विवरण
Class Name	कक्षा का नाम।
Attributes	कक्षा के गुण (बेटा सदस्य)।
Operations	कक्षा के ऑपरेशन (सदस्य फंक्शन)।
Association	कक्षाओं के बीच सामान्य संबंध।
Generalization	एक कक्षा दूसरी की उप-प्रकार है।
Aggregation	"has-a" प्रकार का संबंध (कमजोर)।
Composition	"part-of" प्रकार का संबंध (मजबूत)।
Dependency	एक कक्षा दूसरी पर निर्भर करती है।

8. Activity Diagram के मूल तत्व

तत्व	विवरण
Initial Node	प्रक्रिया की शुरुआत।
Action/Activity	कार्य या गतिविधि।
Decision	निर्णय बिंदु (True/False, Yes/No)।
Fork	एक से अधिक समानांतर गतिविधियाँ शुरू करना।
Join	समानांतर गतिविधियों को मिलाना।
Final Node	प्रक्रिया का अंत।

10. State Machine Diagram के मूल तत्व

तत्व	विवरण
Initial State	प्रारंभिक अवस्था।
State	किसी ऑब्जेक्ट की अवस्था।
Transition	एक अवस्था से दूसरी अवस्था में परिवर्तन।
Self Transition	उसी अवस्था में परिवर्तन।
Final State	अंतिम अवस्था।

12. UML का महत्व

- सॉफ्टवेयर विकास के सभी चरणों में सहायक।
- प्रणाली का बेहतर विश्लेषण और डिजाइन संभव।
- टीम के बीच बेहतर संचार।
- सिस्टम की समझ और दस्तावेजीकरण को आसान बनाता है।
- गुणवत्ता, पुनः उपयोग और रखरखाव को बढ़ाता है।





अध्याय - 19 : MCQs (बहुविकल्पीय प्रश्न)



नोट : नीचे दिए गए प्रश्न System Analysis & Design (SAD) के पूरे पाठ्यक्रम पर आधारित हैं।

1. SAD का मुख्य उद्देश्य क्या है? (A) हार्डवेयर डिजाइन करना (B) सॉफ्टवेयर कोड लिखना (C) उपयोगकर्ता की आवश्यकताओं के अनुसार समाधान प्रदान करना (D) डेटा एंट्री करना	2. निम्न में से कौन-सा SAD का चरण नहीं है? (A) समस्या विश्लेषण (B) डेटा संग्रहण (C) मशीन इंस्टॉलेशन (D) सिस्टम डिजाइन	3. सिस्टम की व्यवहारिकता जाँचने के लिए किस अध्ययन को किया जाता है? (A) फिजिविलिटी स्टडी (B) डेटाबेस स्टडी (C) कोडिंग स्टडी (D) नेटवर्क स्टडी	4. निम्न में से कौन प्रोजेक्ट प्रबंधन का भाग नहीं है? (A) योजना बनाना (B) संसाधन आवंटन (C) कोडिंग (D) निगरानी और नियंत्रण
5. सिस्टम डिजाइन में निम्न में से कौन-सा आउटपुट डिजाइन का घटक नहीं है? (A) रिपोर्ट डिजाइन (B) आउटपुट मीडिया (C) डेटा वैलिडेशन (D) उपयोगकर्ता इंटरफेस	6. निम्न में से कौन-सा डेटाबेस मॉडल है? (A) हायरार्किकल मॉडल (B) नेटवर्क मॉडल (C) रिलेशनल मॉडल (D) सभी उपर्युक्त	7. डेटा नॉर्मलाइजेशन का मुख्य उद्देश्य क्या है? (A) डेटा को तेज़ बनाना (B) डेटा में रिडंडेंसी कम करना (C) रिपोर्ट को आकर्षक बनाना (D) डेटा एंट्री बढ़ाना	8. सिस्टम टेस्टिंग में बग खोजने के लिए कौन-सा टेस्ट किया जाता है? (A) यूनिट टेस्टिंग (B) इंटीग्रेशन टेस्टिंग (C) सिस्टम टेस्टिंग (D) एक्सेप्टेंस टेस्टिंग
9. परिवर्तन प्रबंधन (Change Management) में शामिल नहीं है? (A) प्रभाव विश्लेषण (B) परिवर्तन स्वीकृति (C) परिवर्तन रिजेक्शन (D) परिवर्तन कार्यान्वयन	10. निम्न में से कौन-सा CASE Tools का लाभ है? (A) गुणवत्ता में सुधार (B) विकास समय में वृद्धि (C) लागत में वृद्धि (D) डेटा रिडंडेंसी में वृद्धि	11. UML में संबंधों को दर्शाने के लिए कौन-सा डायग्राम उपयोग होता है? (A) क्लास डायग्राम (B) सीक्वेंस डायग्राम (C) स्टेट मशीन डायग्राम (D) यूज केस डायग्राम	12. यूज केस डायग्राम में कौन-सी आकृति एक्टर को दर्शाती है? (A) (B) (C) (D)
13. क्लास डायग्राम में संबंधों को दर्शाने के लिए कौन-सी लाइन का उपयोग होता है? (A) डैशड लाइन (B) सॉलिड लाइन (C) डबल लाइन (D) डॉटेड लाइन	14. एक्टिविटी डायग्राम में निर्णय (Decision) को किस प्रतीक से दर्शाया जाता है? (A) (B) (C) (D)	15. स्टेट मशीन डायग्राम में प्रारंभिक स्थिति (Initial State) को किस प्रतीक से दर्शाती है? (A) (B) (C) (D)	16. निम्न में से कौन-सा लोअर CASE टूल है? (A) Visual Paradigm (B) Rational Rose (C) Code::Blocks (IDE) (D) StarUML
17. निम्न में से कौन UPPER CASE टूल का उदाहरण है? (A) Rational Rose (B) Visual Studio Code (C) Borland C++ (D) MySQL	18. डिप्लॉयमेंट डायग्राम का मुख्य उपयोग किसके लिए होता है? (A) कोड डिजाइन (B) हार्डवेयर टोपोलॉजी (C) डेटाबेस डिजाइन (D) रिपोर्ट डिजाइन	19. निम्न में से कौन-सा आउटपुट डिजाइन का महत्व है? (A) डेटा संग्रहण (B) उपयोगकर्ता संतुष्टि (C) कोडिंग में सरलता (D) डेटा वैलिडेशन	20. सिस्टम मॉडलिंग के प्रकारों में कौन-सा शामिल नहीं है? (A) करेक्टिव मॉडलिंग (B) एडोप्टिव मॉडलिंग (C) परफेक्टिव मॉडलिंग (D) इंप्लीमेंटेशन मॉडलिंग

उत्तर कुंजी (Answer Key)

- | | | | | |
|---------|---------|---------|---------|---------|
| 1. (C) | 2. (C) | 3. (A) | 4. (C) | 5. (C) |
| 6. (D) | 7. (B) | 8. (D) | 9. (C) | 10. (A) |
| 11. (A) | 12. (A) | 13. (B) | 14. (B) | 15. (A) |
| 16. (C) | 17. (A) | 18. (B) | 19. (B) | 20. (D) |



सफलता मंत्र

- योजना बनाओ, समय पर कार्य करो।
- समस्याओं को समझो, समाधान स्वयं मिलेगा।
- निरंतर अभ्यास ही सफलता की कुंजी है।
- सीखते रहो, बढ़ते रहो, सफल होते जाओ।



नोट : MCQs का अभ्यास आपके ज्ञान को मजबूत बनाता है और परीक्षा में अच्छे अंक दिलाता है। नियमित अभ्यास करें।

